Systemy wbudowane

dr hab. inż. Dariusz Makowski, prof. uczelni

Katedra Mikroelektroniki i Technik Informatycznych

tel. 631 2720

dmakow@dmcs.pl

http://fiona.dmcs.pl/sw

Sprawy formalne

- Informacje wstępne
- Egzamin
- Laboratorium
- Materiały do wykładu

Ogłoszenie

Research Week

- https://researchweek.p.lodz.pl/o-research-week
- **Research Week** rozpocznie się 10 marca i potrwa do 16 marca 2023 r.
- Zwolnienie w wykładu SW w dniu 10 marca
- Spotkanie o godz 8.30 w auli im. T. Paryjczaka w Alchemium

  E2TOP - czyli jak zdobyć naukowy szczyt?
  prof. Łukasz Szymański, przewodniczący rady programowej E^{2}TOP

Zakres przedmiotu

- Laboratorium
- Systemy mikroprocesorowe, systemy wbudowane
- Rodzina procesorów ARM
- Urządzenia peryferyjne
- Programy wbudowane na przykładzie procesorów ARM
- Metodyki projektowania systemów wbudowanych
- Interfejsy w systemach wbudowanych
- Systemy czasu rzeczywistego

Zakres przedmiotu

- Laboratorium
- Systemy mikroprocesorowe, systemy wbudowane
- Rodzina procesorów ARM
- Urządzenia peryferyjne
- Programy wbudowane na przykładzie procesorów ARM
- Metodyki projektowania systemów wbudowanych
- Interfejsy w systemach wbudowanych
- Systemy czasu rzeczywistego

Laboratorium

Adres:

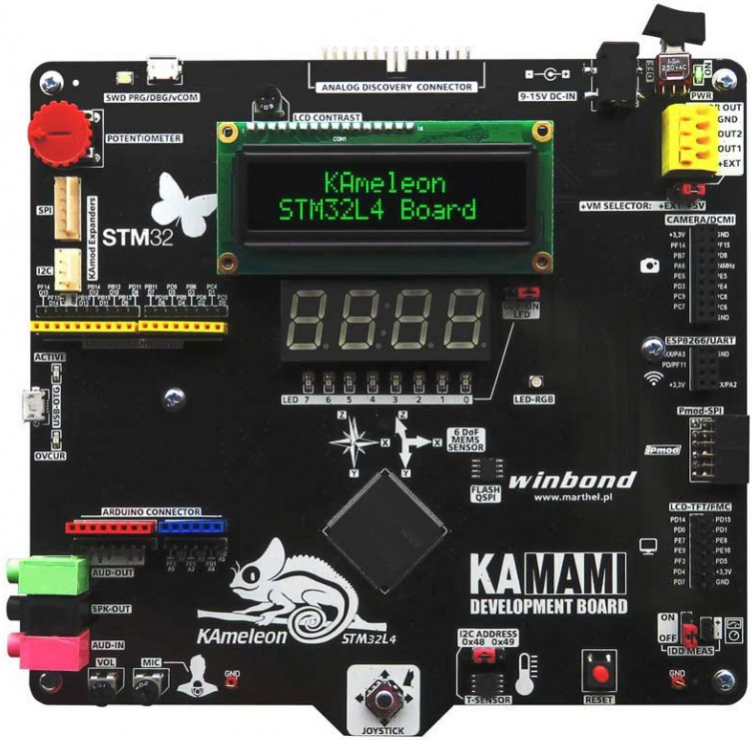
- Budynek B18, laboratorium M

Praktyczne ćwiczenia z użyciem procesora ARM:

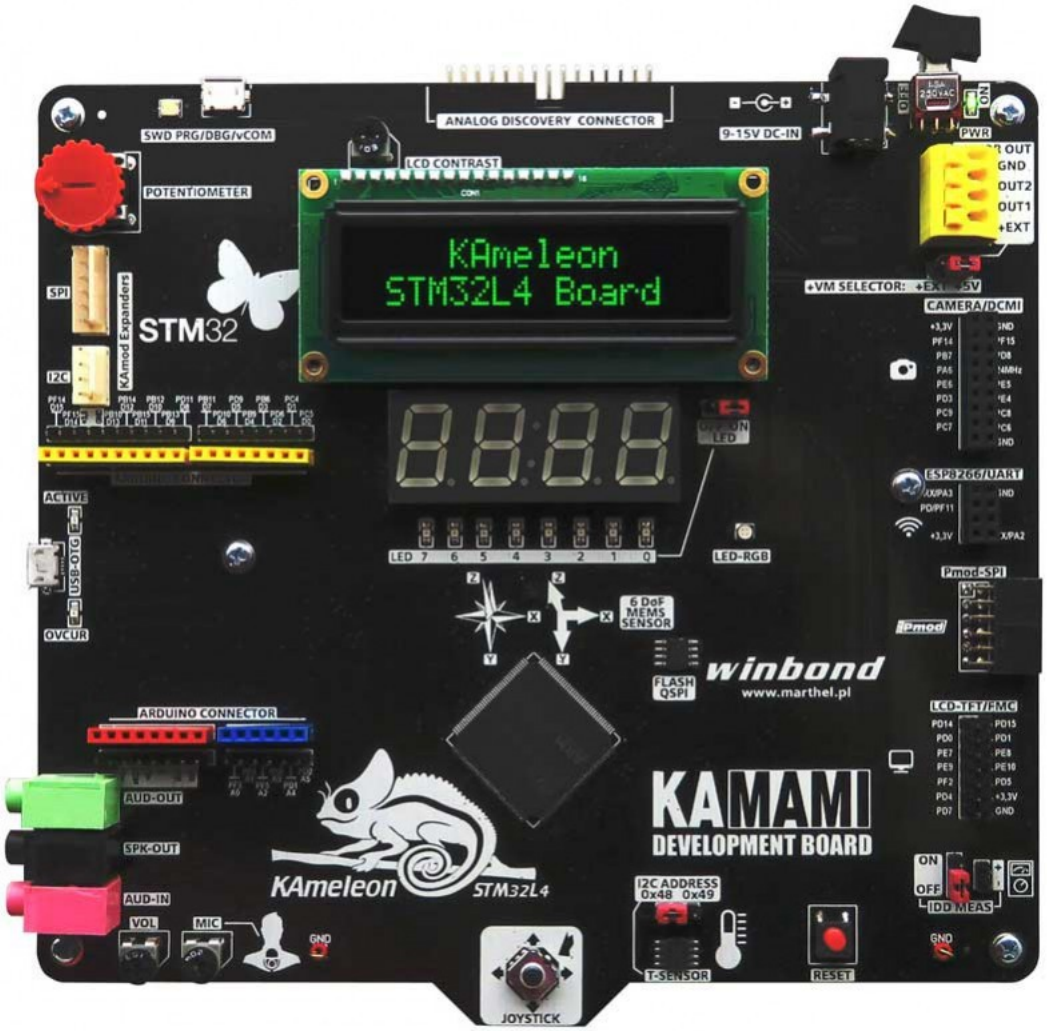
- System operacyjny Windows
- Środowisko deweloperskie: STM32CubeIDE
- Narzędzia GNU
- Zestaw ewaluacyjny KAmeleon-STM32L4
- Urządzenia peryferyjne
- System czasu rzeczywistego.

Sprzęt – STM32 Evaluation Module (1)

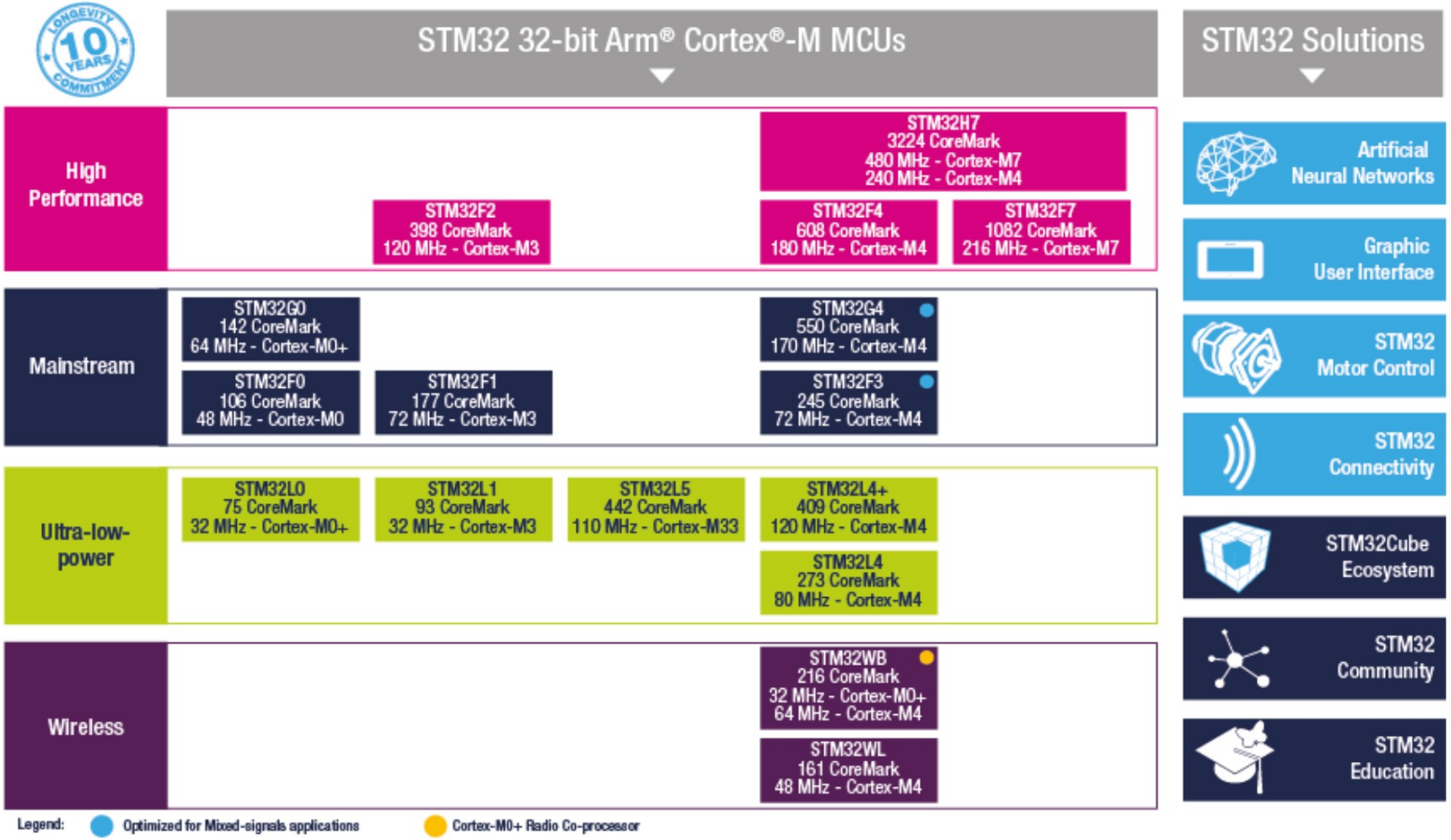
- **Mikrokontroler z rdzeniem ARM Cortex M4:** STM32L496ZGT6
- **Pamięci:** 320 kB SRAM,1 MB FLASH, 1 MB SPI-PROM
- **Interfejsy:** SPI, I2C, USB-OTG, EIA RS232
- **Wyświetlacz:** 4 cyfry LED i wyświetlacz alfa-numeryczny, 8x LED, RGB LED
- **Audio:** mikrofon, wzm. udio
- **Urządzenia peryferyjne:** sterownik silnika DC , termometr, akcelerometr, magnetometr
- **Interfejs (debuger, programator):** Serial Wire Debug
- **Złącza:** DCMI (kamera), WiFi, PMOD, etc.
- **Programator:** ST-Link
- **Oznaczenie producenta:** KAmeleon-STM32L4



Sprzęt – STM32 Evaluation Module (2)



Rodzina 32-bits Mikrokontrolerów STM32



https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

Rodzina 32-bits Mikrokontrolerów STM32 (2)

STM32 Family	Cortex-M	Thumb	Thumb-2	Multiply in Hardware	Divide in Hardware	Saturated math	DSP	FPU	ARM Architecture
F0	M0	Most	Some	32-bit result	No	No	No	No	ARMv6-M
L0	M0+	Most	Some	32-bit result	No	No	No	No	ARMv6-M
F1, F2, L1	M3	Entire	Entire	32/64-bit result	Yes	Yes	No	No	ARMv7-M
F3, F4, L4	M4	Entire	Entire	32/64-bit result	Yes	Yes	Yes	Yes SP	ARMv7E-M
F7	M7	Entire	Entire	32/64-bit result	Yes	Yes	Yes	Yes SP & DP	ARMv7E-M

STM32 Family	Cortex-M	SysTick Timer	Bit-Banding	Memory Protection Unit (MPU)	CPU Cache	OS Support	Memory Architecture
F0	M0	Yes	Yes	No	No	Yes	Von Neumann
L0	M0+	Yes	Yes	Yes	No	Yes	Von Neumann
F1, F2, L1	M3	Yes	Yes	Yes	No	Yes	Harvard
F3, F4, L4	M4	Yes	Yes	Yes	No	Yes	Harvard
F7	M7	Yes	No	Yes	Yes	Yes	Harvard

STM32L4 Ultra-low-power Series Microcontrollers

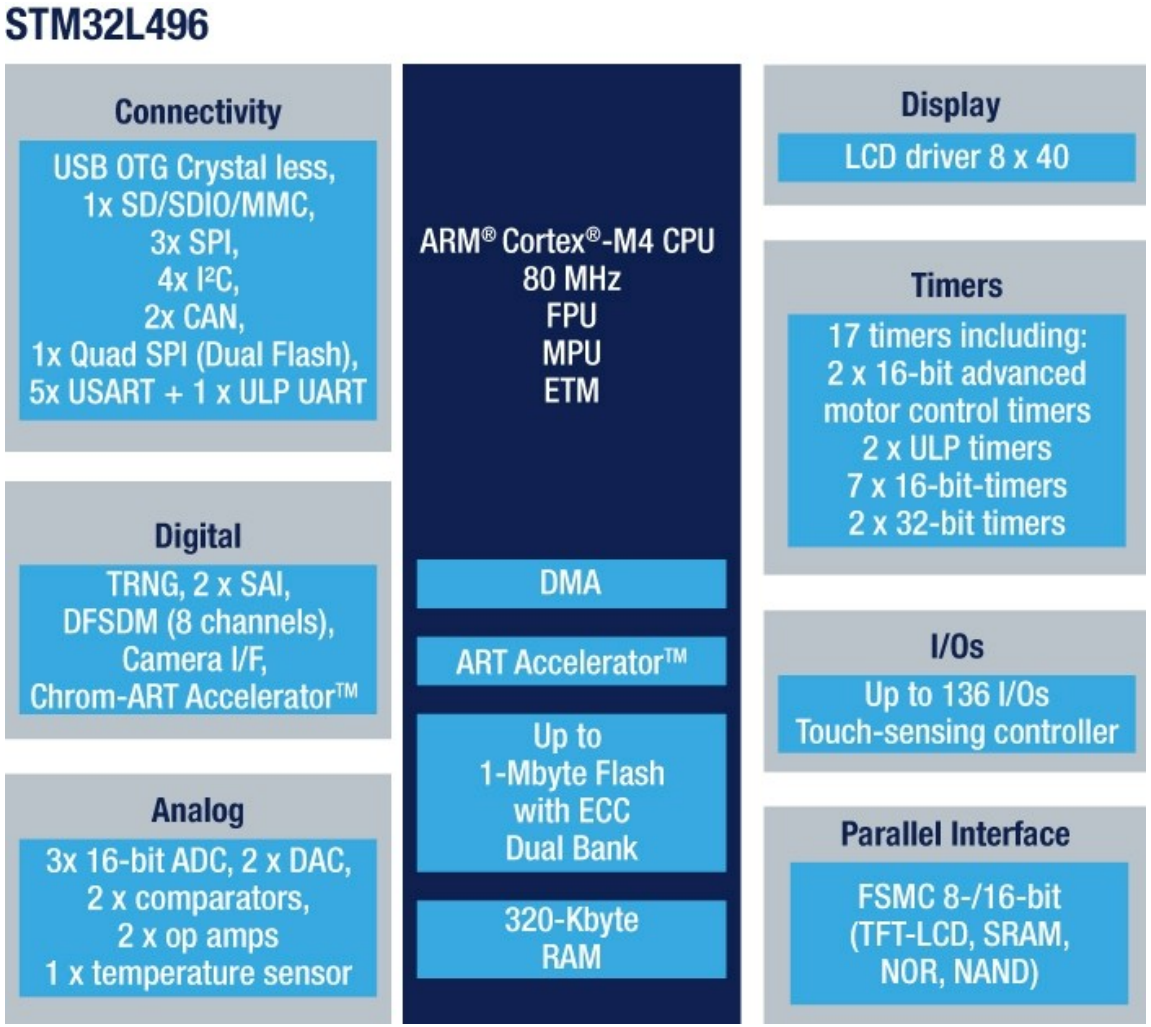
Arm® Cortex®-M4 (DSP + FPU) – 80 MHz

- ART Accelerator™
- USART, SPI, I²C
- Quad-SPI
- 16- and 32-bit timers
- SAI + audio PLL
- SWP
- 2x CAN
- 2x 12-bit DACs
- Temperature sensor
- Low voltage 1.71 to 3.6 V
- V_{BAT} mode
- Unique ID
- Capacitive touch sensing
- AES-128/256* and SHA-256**

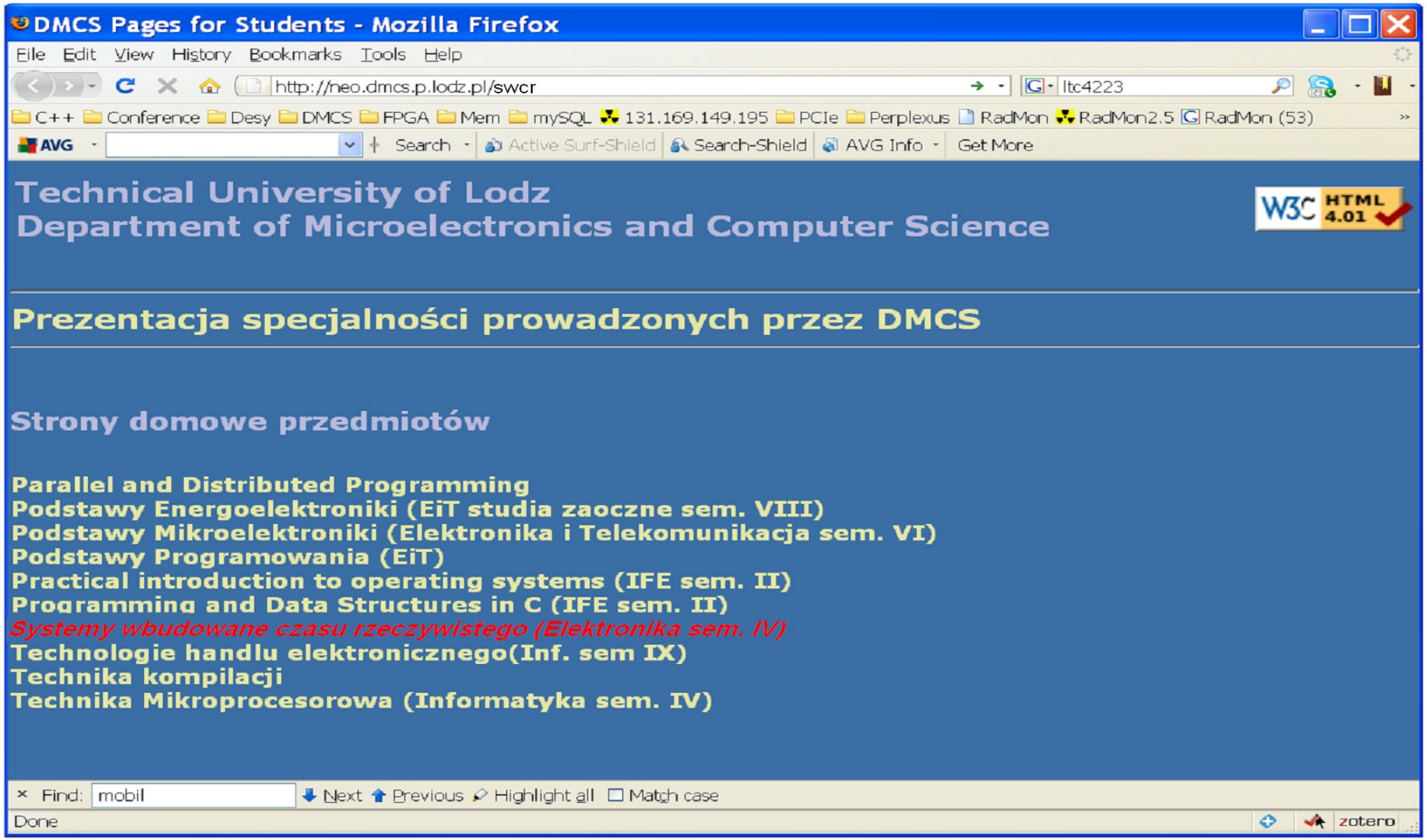
STM32 L4 Product line	Flash (KB)	RAM (KB)	Memory I/F FSMC	Op-Amp	CAN	Sigma Delta Interface	12-bit ADC 5 Msps 16-bit HW oversampling	DAC	SAI	USB2.0 OTG FS	USB Device	Segment LCD driver	Chrom-ART Accelerator™
STM32L4x6 - USB OTG + Segment LCD Lines													
STM32L496**	512 to 1024	320	•	2	2	8x ch	3	2	2	•		Up to 8x40	•
STM32L476*	256 to 1024	128	•	2	1	8x ch	3	2	2	•		Up to 8x40	
STM32L4x5 - USB OTG lines													
STM32L475	256 to 1024	128	•	2	1	8x ch	3	2	2	•			
STM32L4x3 - USB Device + Segment LCD lines													
STM32L433*	128 to 256	64		1	1		1	2	1		•	Up to 8x40	
STM32L4x2 - USB Device lines													
STM32L452*	256 to 512	160		1	1	4x ch	1	1	1		•		
STM32L432*	128 to 256	64		1	1		1	2	1		•		
STM32L412*	64 to 128	40		1			2				•		
STM32L4x1 - Access lines													
STM32L471	512 to 1024	128	•	2	1	8x ch	3	2	2				
STM32L451	256 to 512	160		1	1	4x ch	1	1	1				
STM32L431	128 to 256	64		1	1		1	2	1				

https://www.st.com/content/st_com/en/products/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/stm32-ultra-low-power-mcus/stm32l4-series.html

Mikrokontroler STM32L496 ...



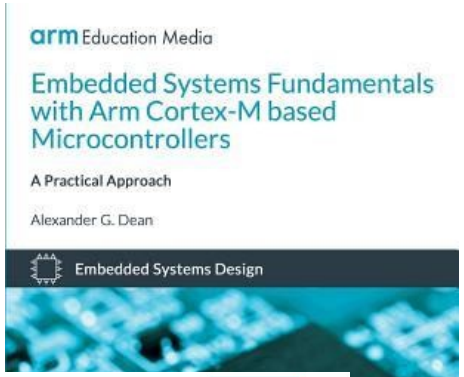
Strona przedmiotu – materiały do wykładu

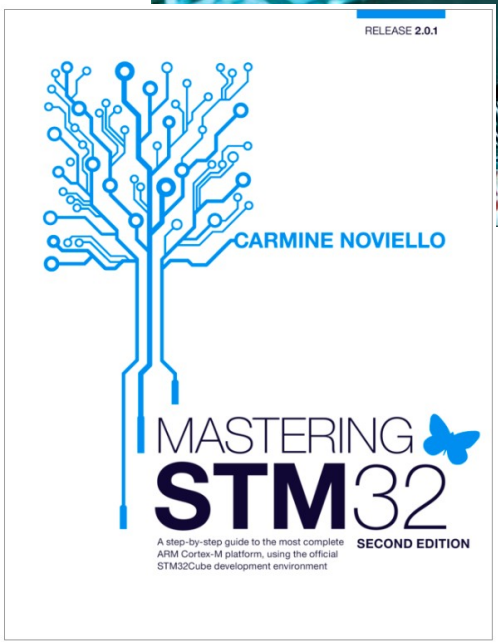


Literatura

Literatura obowiązkowa:

- **Materiały wykładowe i laboratoryjne**
- A. Dean, “Embedded Systems Fundamentals with ARM Cortex-M based Microcontrollers: A Practical Approach”, ARM Education Media, 2017, ISBN-13 9781911531012, ISBN-10 1911531018
- C. Noviello, “Mastering STM32 - Second Edition”, Leanpub 2022
- STM32 MCU Developer Resources
  - https://www.st.com/content/st_com/en/stm32-mcu-developer-zone/developer-resources.html
  - https://www.st.com/resource/en/user_manual/dm00173145-description-of-stm32l4l4-hal-and-lowlayer-drivers-stmicroelectronics.pdf





Zakres przedmiotu

- Laboratorium
- Systemy mikroprocesorowe, systemy wbudowane
- Rodzina procesorów ARM
- Urządzenia peryferyjne
- Pamięci i dekodery adresowe
- Programy wbudowane na przykładzie procesorów ARM
- Metodyki projektowania systemów wbudowanych
- Interfejsy w systemach wbudowanych
- Systemy czasu rzeczywistego

Definicje podstawowe (1)

- **Procesor (ang. Processor, Central Processing Unit)**

  Urządzenie cyfrowe, sekwencyjne, potrafiące pobierać dane z pamięci, interpretować je i wykonywać jako rozkazy

- **Mikroprocesor (ang. Microprocessor)**

  Układ cyfrowy wykonany jako pojedynczy układ scalony o wielkim stopniu integracji (VLSI) zdolny do wykonywania operacji cyfrowych według dostarczonych mu informacji, np.: x86, Z80, 68k

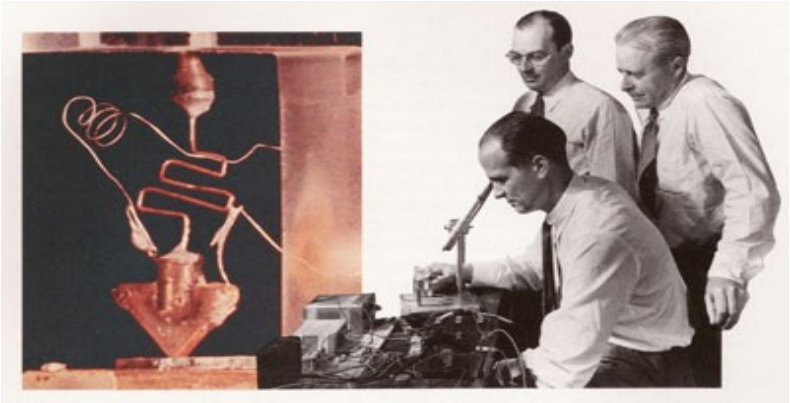
- **Mikrokontroler (ang. Microcontroller)**

  Komputer wykonany w jednym układzie scalonym, używany do sterowania urządzeniami elektronicznymi. Oprócz jednostki centralnej CPU posiada zintegrowane pamięci oraz urządzenia peryferyjne, np.: Intel 80C51, Atmel Atmega128, Freescale MCF5282, ARM926EJ-S

Historia mikroprocesorów (1)

1940 – Russell Ohl – demonstracja złącza półprzewodnikowego (dioda germanowa, bateria słoneczna)

1947 – Shockley, Bardeen, Brattain prezentują pierwszy tranzystor



Pierwszy tranzystor, Bell Laboratories

Pierwszy układ scalony, TI

1958 – Jack Kilby wynalazł pierwszy układ scalony

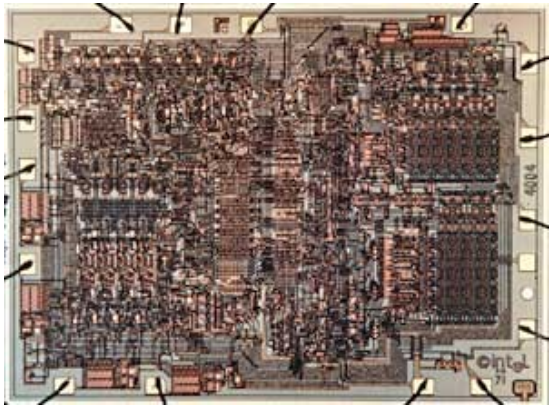
1967 – Laboratorium Fairchild oferuje pierwszą pamięć nieulotną ROM (64 bity)

1969 – Noyce i Moore opuszczają laboratorium Fairchild, powstaje niewielka firma INTEL. INTEL produkuje pamięci SRAM (64 bity). Japońska firma Busicom zamawia 12 różnych układów realizujących funkcje kalkulatorów.

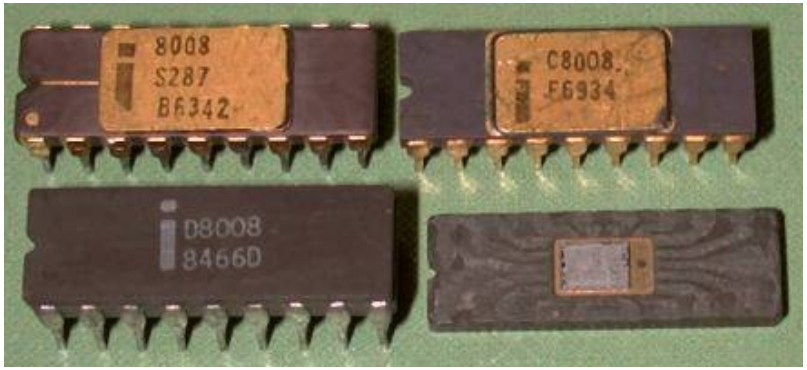
Historia mikroprocesorów (2)

1970 - **F14 CADC** (Central Air Data Computer) mikroprocesor zaprojektowany przez Steve'a Gellera i Raya Holta na potrzeby armii amerykańskiej (myśliwiec F-14 Tomcat)

1971 - **Intel 4004** 4-bitowy procesor realizujące funkcje programowalnego kalkulatora **(**powszechnie uznaje się za pierwszy na świecie mikroprocesor), 3200 tranzystorów. INTEL wznawia pracę nad procesorami, Faggin z Fairchild pomaga rozwiązać problemy.



Zdjęcie 4-bitowego procesora INTEL 4004



8-bitowe procesory INTEL-a

1972 – Faggin rozpoczyna prace nad 8-bitowym procesorem INTEL 8008. Rynek zaczyna się interesować układami “programowalnymi” - procesorami.

Historia mikroprocesorów (3)

1974 – INTEL wprowadza na rynek ulepszona wersję 8008, procesor Intel 8080. Faggin opuszcza firmę Intel i zakłada firmę o nazwie Zilog. Motorola oferuje 8-bitowy procesor 6800 (NMOS, 5 V).

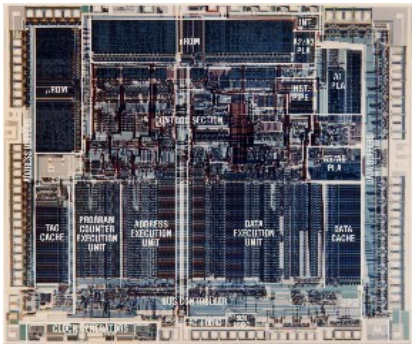
1975 – 8-bitowy procesor INTEL 6502 (technologia MOS) – najtańszy proc. na rynku.

1978 – Pierwszy 16-bitowy procesor 8086 (ulepszony 8080).

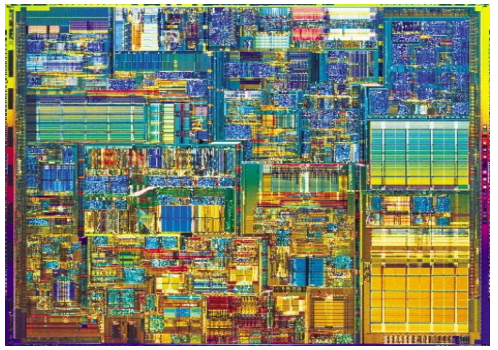
1979 – Motorola oferuje 16-bitowy procesor 68000.

1980 – Motorola wprowadza nowy 32-bitowy procesor 68020, 200,000 tranzystorów.

.....



Motorola 68020



Intel, Pentium 4 Northwood

Intel 386, 486, Pentium I, II, III, IV, Centrino, Pentium D, Duo/Quad core, ...

Motorola 68030, 68040, 68060, PowerPC, ColdFire, ARM 7, ARM 9, StrongARM, ...

Definicje podstawowe (2)

- **Komputer (ang. Computer)**

  Urządzenie elektroniczne, maszyna cyfrowa zdolna do przetwarzania danych cyfrowych zgodnie z dostarczonym programem

- **Komputer (system) wbudowany (ang. Embedded Computer)**

  Dedykowany system komputerowy, niewielkich rozmiarów sterownik *wbudowany w urządzenie*, przeznaczony do sterowania urządzeniem mechanicznym, elektrycznym lub elektronicznym

- **Komputer osobisty (ang. Personal Computer)**

  System komputery przeznaczony do użytku osobistego, domowego lub biurowego. Komputer wyposażony w system operacyjny przeznaczony do wykonywania *aplikacji wykorzystywanych przez użytkownika*

- **Architektura komputera (ang. Computer Architecture)**

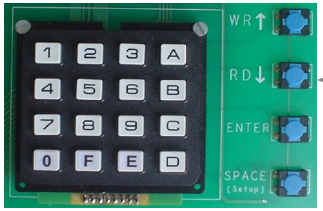
  - Sposób organizacji oraz współpracy podstawowych elementów systemu komputerowego, tj. procesora, pamięci oraz urządzeń peryferyjnych
  - Opis komputera z punktu widzenia programisty w języku niskiego poziomu (asembler). Budowa procesora, potoku wykonawczego oraz model programowy procesora

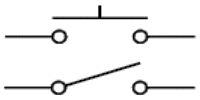
Komputer wbudowany (embedded computer)

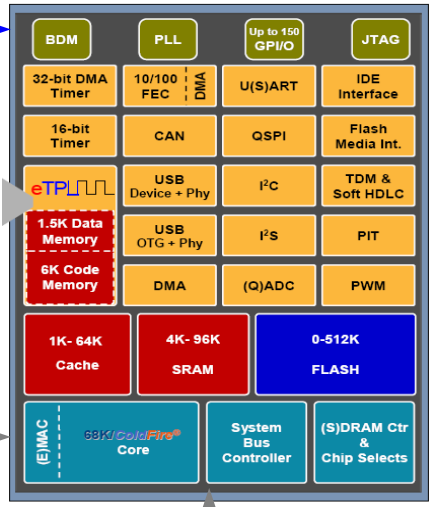
Komunikacja z komputerem zewnętrznym

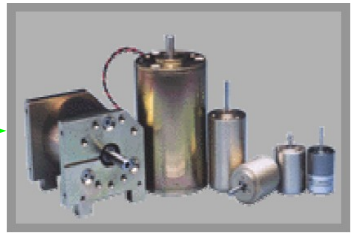
Czujniki, np. czujnik temperatury, obrotów, itd

Kamera



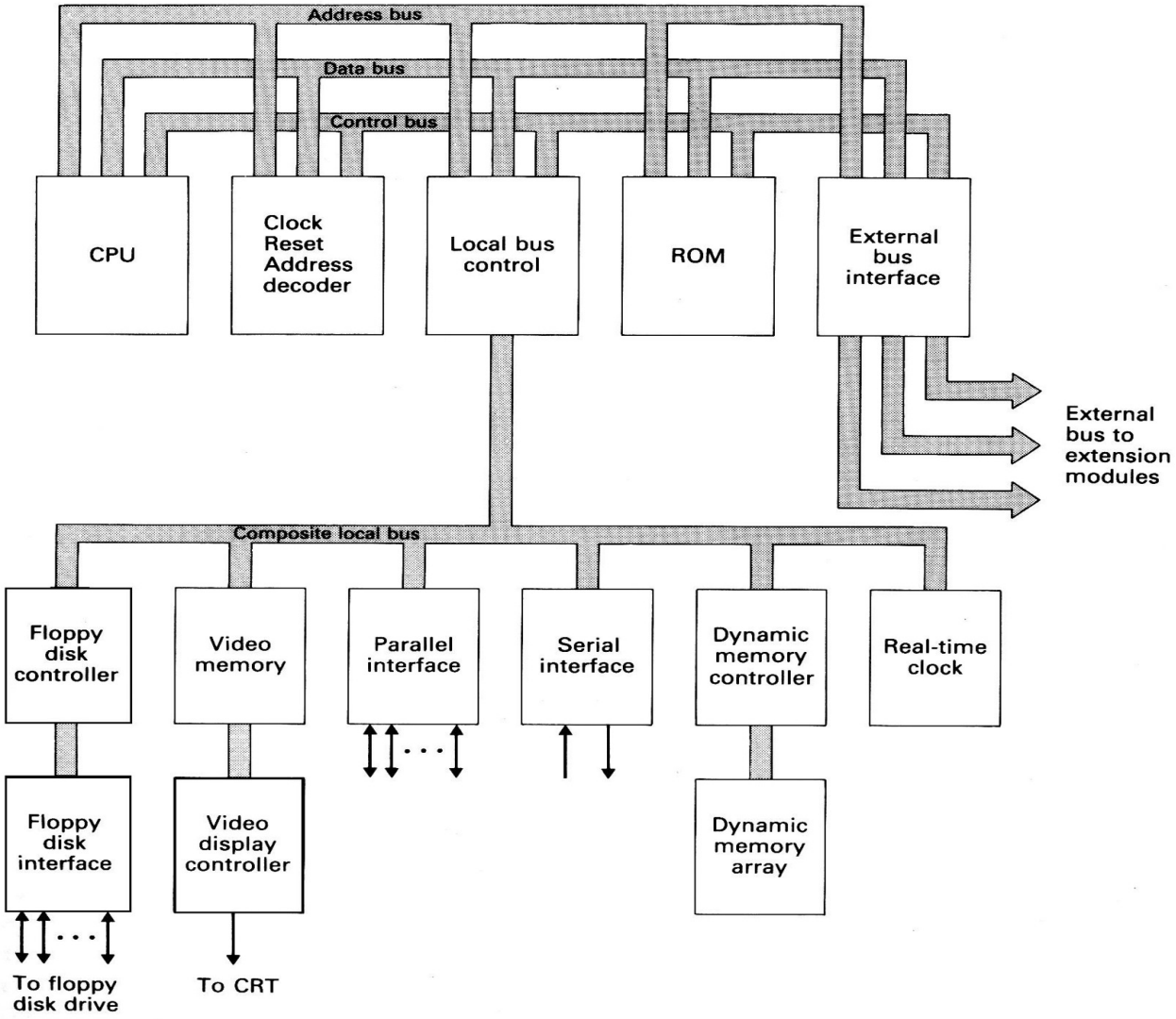




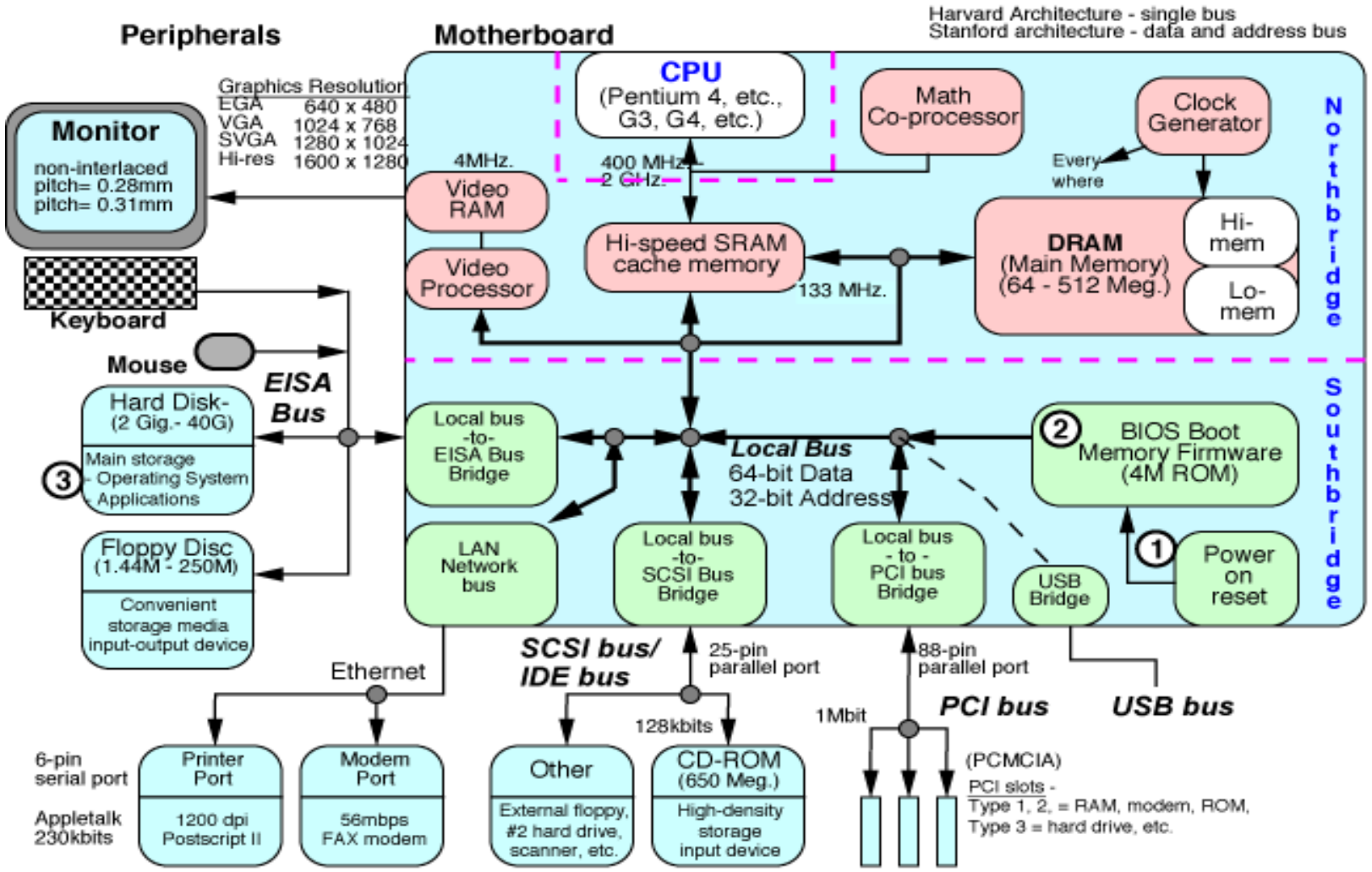


Elementy wykonawcze, np. silniki, przekaźniki

Komputer uniwersalny



Schemat blokowy komputera osobistego



Definicje podstawowe (3)

- **Pamięć komputerowa (ang. Computer Memory)**
  Pamięć komputerowa to urządzenie elektroniczne lub mechaniczne służące do przechowywania danych i programów (systemu operacyjnego oraz aplikacji).
- **Urządzenia zewnętrzne, peryferyjne (ang. Peripheral Device)**
  Urządzenia elektroniczne dołączone do procesora przez magistrale systemową lub interfejs. Urządzenia zewnętrzne wykorzystywane są do realizowania specjalizowanej funkcjonalności systemu.
- **Magistrala (ang. bus)**
  Połączenie elektryczne umożliwiające przesyłanie danym pomiędzy procesorem, pamięcią i urządzeniami peryferyjnymi. Magistra systemowa zbudowane jest zwykłe z kilkudziesięciu połączeń elektrycznych (ang. Parallel Bus) lub szeregowego połączenia (ang. Serial Bus).
- **Interface (ang. Interface)**
  Urządzenie elektroniczne lub optyczne pozwalające na komunikację między dwoma innymi urządzeniami, których bezpośrednio nie da się ze sobą połączyć.

Definicje podstawowe (4)

- **Komputer SoC (ang. System-on-Chip)**

  Układ scalony wielkiej integracji zawierający ***kompletny system elektroniczny zintegrowany z układami analogowymi, cyfrowo-analogowymi oraz radiowymi.*** Poszczególne moduły tego systemu, ze względu na ich złożoność, pochodzą zwykle od różnych dostawców, np. rdzeń procesora od jednego producenta, układy peryferyjne od innego, interfejsy od jeszcze innego, itd...

  Typowym obszarem zastosowań SoC są systemy wbudowane, a najbardziej rozpowszechnionym przedstawicielem tego rozwiązania są systemy oparte na procesorach ARM.

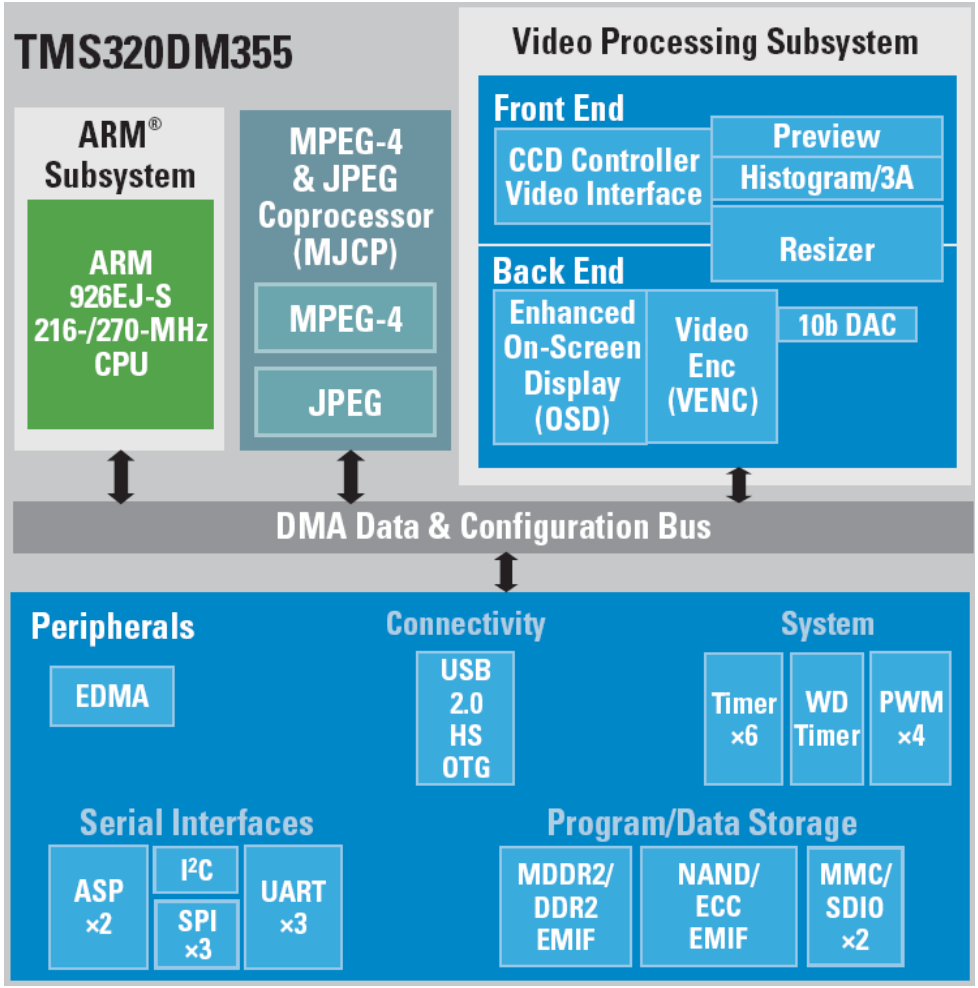
  W przypadku, gdy nie jest możliwa integracja wszystkich układów na jednym podłożu półprzewodnikowym, poszczególne moduły wykonuje się na osobnych kryształach, a całość zamyka się w jednej obudowie, SiP (ang. System-in-a-package).

  SoC różnią się od mikrokontrolerów ***znacznie wydajniejszą jednostka obliczeniową CPU*** (pozwalającej uruchamiać systemy operacyjne, np. Linux, Windows) oraz są zwykłe ***wyposażone w specjalizowane układy peryferyjne.***

SoC - DaVinci, digital media processor

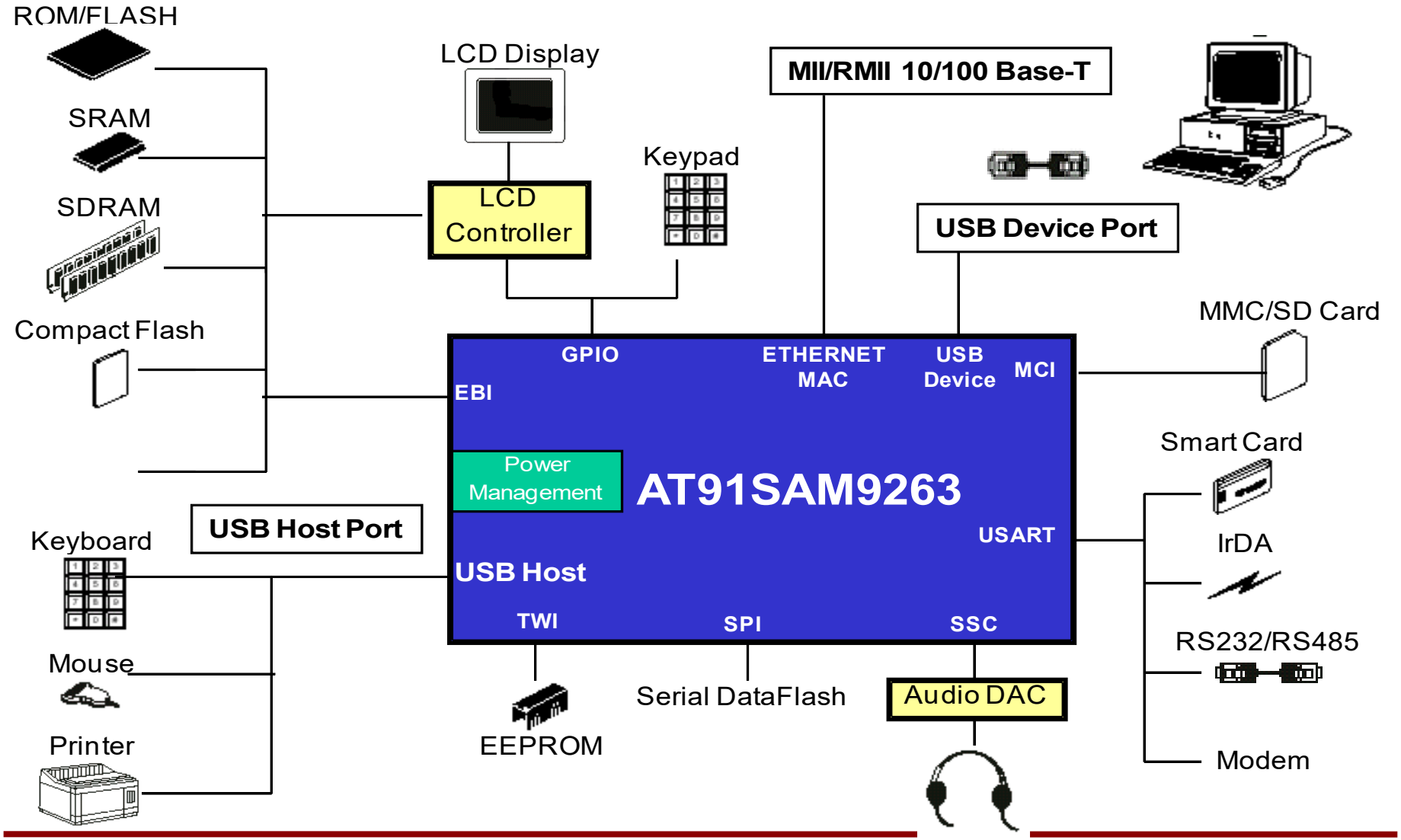
DaVinci DM355

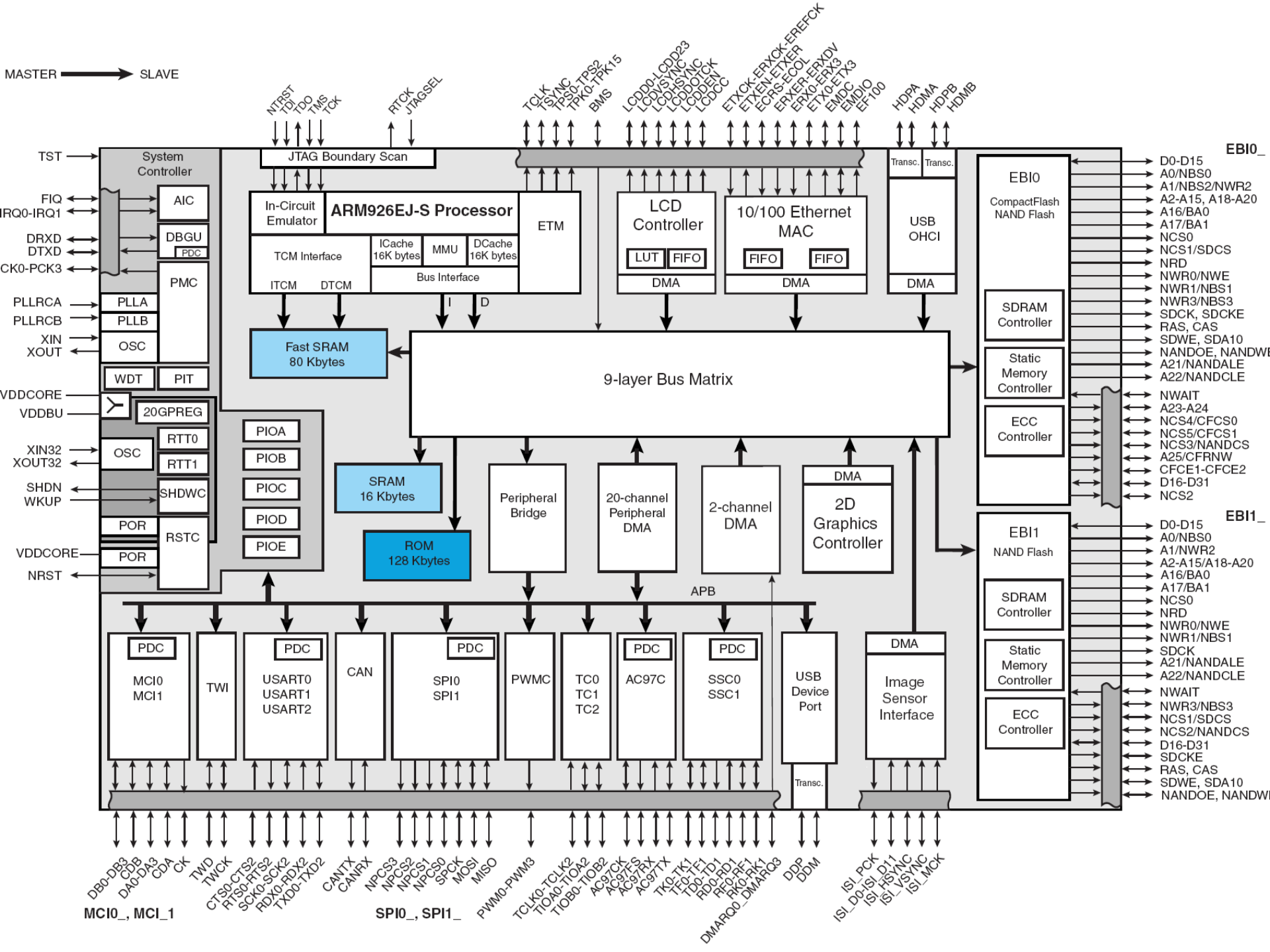
- SoC opracowany przez firmę Texas Instruments
- Dedykowany co-procesor do przetwarzania dźwięku i obrazu w czasie rzeczywistym
- Niski pobór energii 400 mW podczas dekodowania HD MPEG4, 1 mW w stanie czuwania (systemy przenośne)
- Bogate interfejsy i układy peryferyjne (sterownik HDD, SD/MMC, USB, Ethernet,...)



Źródło: www.ti.com

Mikrokontroler AT91SAM9263 (3)




MASTER
SLAVE
NTRST
TDI
TDO
TMS
TCK
RTCK
JTAGSEL
TCLK
TSYNC
TPS0-TPS2
TPK0-TPK15
BMS
LCDD0-LCDD23
LCDVSYNC
LCDHSYNC
LCDDOTCK
LCDDEN
LCDCC
ETXCK-ERXCK-EREFCK
ETXEN-ETXER
ECRS-ECOL
ERXER-ERXDV
ERX0-ERX3
ETX0-ETX3
EMDC
EMDIO
EF100
HDPA
HDMA
HDPB
HDMB
TST
System Controller
JTAG Boundary Scan
FIQ
IRQ0-IRQ1
AIC
DRXD
DTXD
DBGU
PDC
CK0-PCK3
PMC
PLLRCA
PLLRCB
PLLA
PLLB
XIN
XOUT
OSC
WDT
PIT
VDDCORE
VDDBU
20GPREG
XIN32
XOUT32
OSC
RTT0
RTT1
SHDN
WKUP
SHDWC
POR
RSTC
VDDCORE
POR
NRST
In-Circuit Emulator
ARM926EJ-S Processor
ETM
TCM Interface
ITCM
DTCM
ICache 16K bytes
MMU
DCache 16K bytes
Bus Interface
I
D
LCD Controller
LUT
FIFO
DMA
10/100 Ethernet MAC
FIFO
FIFO
DMA
Transc.
Transc.
USB OHCI
DMA
Fast SRAM 80 Kbytes
9-layer Bus Matrix
PIOA
PIOB
PIOC
PIOD
PIOE
SRAM 16 Kbytes
ROM 128 Kbytes
Peripheral Bridge
20-channel Peripheral DMA
2-channel DMA
DMA
2D Graphics Controller
APB
EBI0
CompactFlash NAND Flash
SDRAM Controller
Static Memory Controller
ECC Controller
EBI0_
D0-D15
A0/NBS0
A1/NBS2/NWR2
A2-A15, A18-A20
A16/BA0
A17/BA1
NCS0
NCS1/SDCS
NRD
NWR0/NWE
NWR1/NBS1
NWR3/NBS3
SDCK, SDCKE
RAS, CAS
SDWE, SDA10
NANDOE, NANDWE
A21/NANDALE
A22/NANDCLE
NWAIT
A23-A24
NCS4/CFCS0
NCS5/CFCS1
NCS3/NANDCS
A25/CFRNW
CFCE1-CFCE2
D16-D31
NCS2
EBI1
NAND Flash
SDRAM Controller
Static Memory Controller
ECC Controller
EBI1_
D0-D15
A0/NBS0
A1/NWR2
A2-A15/A18-A20
A16/BA0
A17/BA1
NCS0
NRD
NWR0/NWE
NWR1/NBS1
SDCK
A21/NANDALE
A22/NANDCLE
NWAIT
NWR3/NBS3
NCS1/SDCS
NCS2/NANDCS
D16-D31
SDCKE
RAS, CAS
SDWE, SDA10
NANDOE, NANDWE
PDC
MCI0
MCI1
TWI
PDC
USART0
USART1
USART2
CAN
PDC
SPI0
SPI1
PWMC
TC0
TC1
TC2
PDC
AC97C
PDC
SSC0
SSC1
USB Device Port
Transc.
DMA
Image Sensor Interface
DB0-DB3
CDB
DA0-DA3
CDA
CK
TWD
TWCK
CTS0-CTS2
RTS0-RTS2
SCK0-SCK2
RDX0-RDX2
TXD0-TXD2
CANTX
CANRX
NPCS3
NPCS2
NPCS1
NPCS0
SPCK
MOSI
MISO
PWM0-PWM3
TCLK0-TCLK2
TIOA0-TIOA2
TIOB0-TIOB2
AC97CK
AC97FS
AC97RX
AC97TX
TK0-TK1
TF0-TF1
TD0-TD1
RD0-RD1
RF0-RF1
RK0-RK1
DMARQ0_DMARQ3
DDP
DDM
ISI_PCK
ISI_D0-ISI_D11
ISI_HSYNC
ISI_VSYNC
ISI_MCK
MCI0_, MCI_1
SPI0_, SPI1_

GNU Tools for ARM processors

GNU ARM toolchain – zestaw narzędzi programistycznych ddla procesorów ARM dostępnych na zasadach licencji GNU GPL (General Public License).

Dostępne narzędzia (***http://www.gnuarm.org/***):

- GCC toolchain (Windows, Linux):
  - Compiler: **gcc-arm-elf**
  - Useful tools: **binutils-2.19**
  - Library C/C++: **newlib-1.16**
  - Debugger GDB
- Windows:
  - **STMCubeIDE** (ver. 1.12.0) - Integrated Development Environment for STM32
  - Środowisko bazuje na frameworku Eclipse/CDT, narzędziach GCC, GDB
  - Biblioteka HAL (Hardware Abstraction Layer)

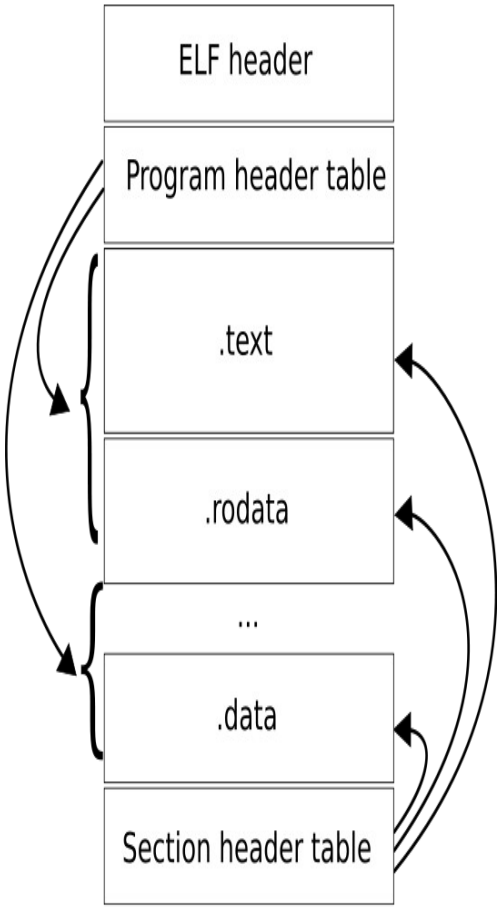
    https://www.st.com/en/development-tools/stm32cubeide.html

COFF vs ELF

- **COFF (Common Object File Format) –** standard plików wykonywalnych, relokowalnych i bibliotek dynamicznych opracowany na potrzeby systemów operacyjnych bazujących na systemie Unix. COFF miał zastąpić standard plików **a.out**. Wykorzystywany na różnych systemach, również Windows. Obecnie standard COFF wypierany jest przez pliki ELF.

- **ELF (Executable and Linkable Format) –** standard plików wykonywalnych, relokowalnych, bibliotek dynamicznych i zrzutów pamięci wykorzystywany na różnych komputerach i systemach operacyjnych, np.: rodzina x86, PowerPC, OpenVMS, BeOS, konsole PlayStation Portable, PlayStation 2, PlayStation 3, Wii, Nintendo DS, GP2X, AmigaOS 4 oraz Symbian OS v9.

- **Przydatne narzędzia:**
  - readelf
  - elfdump
  - objdump



Źródło: wikipedia

Debuger GDB

arm-elf-gdb <nazwa pliku.elf>

run – uruchomienie programu (załadowanie i uruchomienie), load – ładuje program

c (continue) – kontynuacja wykonywania programu

b (breakpoint) – ustawienie pułapki, np. b 54, b main, b sleep

n (next) – wykonanie funkcji (bez zagłębiania się do jej wnętrza)

s (step) – wykonanie funkcji, zatrzymuje się wewnątrz funkcji

d (display) – wyświetlenie zmiennej/rejestru, disp Counter

p (print) – wyświetlenie (jednorazowe) zmiennej/rejestru

x – wyświetlenie obszaru pamięci **x/10x 0xFFFF.F000**

i (info) – wyświetla informacje o pułapkach, rejestrach, etc...

Modyfikatory:

/x – wyświetla w postaci szesnastkowej

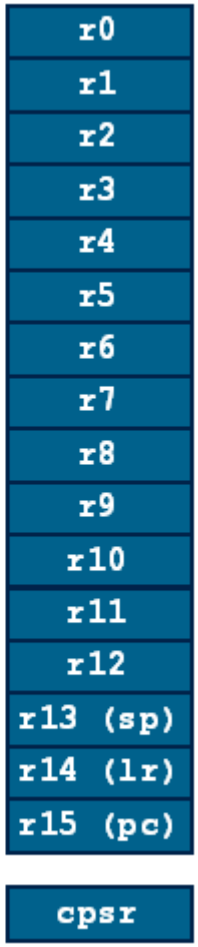
/t - wyświetla w postaci binarnej

/d - wyświetla w postaci dziesiętnej

Rejestry procesora a GDB

(gdb) info r

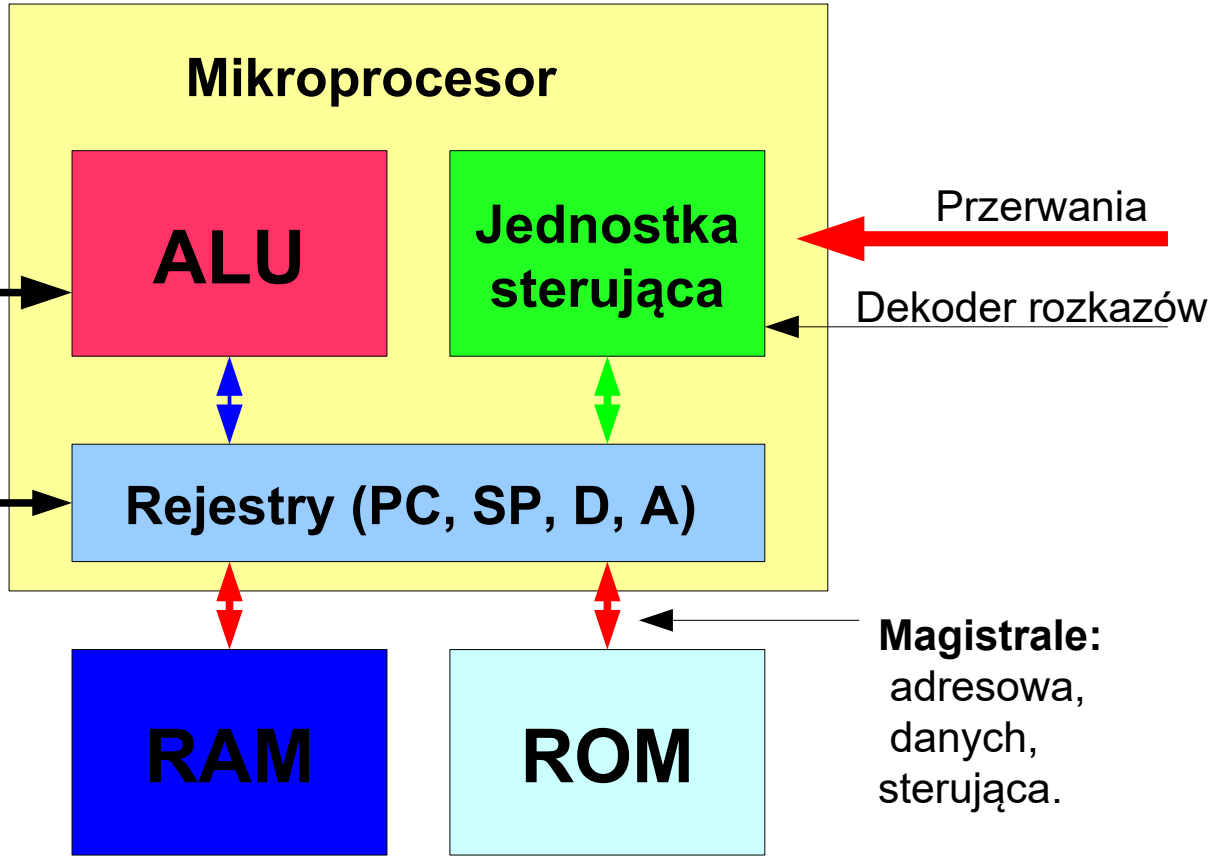
r0	0x2	0x2
r1	0x20000ba4	0x20000ba4
r2	0x57b	0x57b
r3	0x270f	0x270f
r4	0x300069	0x300069
r5	0x3122dc	0x3122dc
r6	0x1000	0x1000
r7	0x800bc004	0x800bc004
r8	0x3122c4	0x3122c4
r9	0x407c81a4	0x407c81a4
r10	0x441029ab	0x441029ab
r11	0x313f2c	0x313f2c
r12	0x313f30	0x313f30
sp	0x313f18	0x313f18
lr	0x20000a7c	0x20000a7c
pc	0x20000474	0x20000474 <delay+60>
fps	0x0	0x0
cpsr	0x80000053	0x80000053



Mikroprocesor

Mikroprocesor to układ cyfrowy wykonany jako pojedynczy układ scalony o wielkim stopniu integracji zdolny do wykonywania operacji cyfrowych według dostarczonych mu instrukcji.

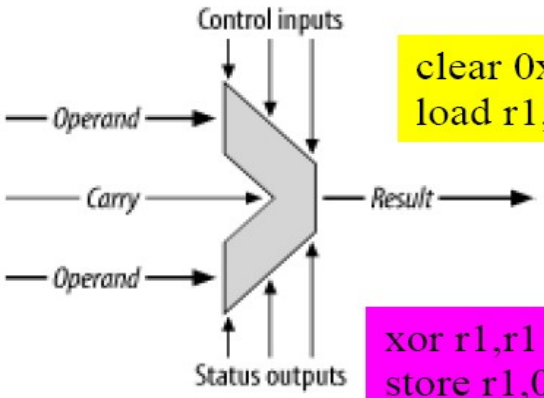
Jednostka arytmetyczno-logiczna (ang. Arithmetic Logic Unit), realizuje podstawowe operacje matematyczne 8, 16, 32, 64-bit



Rejestry procesora - obszar (plik) rejestrów (ang. registers file) - komórki szybkiej pamięci statycznej, umieszczonej, wewnątrz procesora, 8, 16, 32, 64, 128-bit,

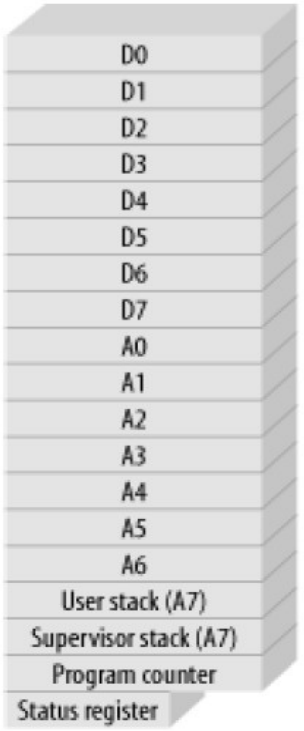
Architektura procesora (2)

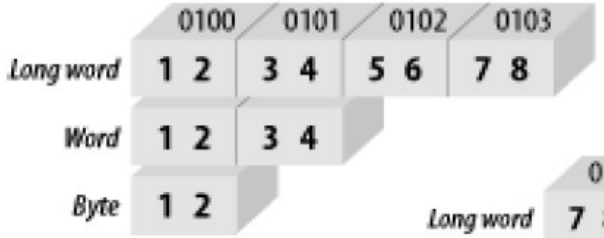
- ALU
  - lista operacji
- zestaw rejestrów
  - A, I, PC, SR, SP,...
- lista rozkazów
  - CISC, RISC
- tryby adresowania
  - R, A, I,...
- przerwania
  - hardware, software
- big/little endian

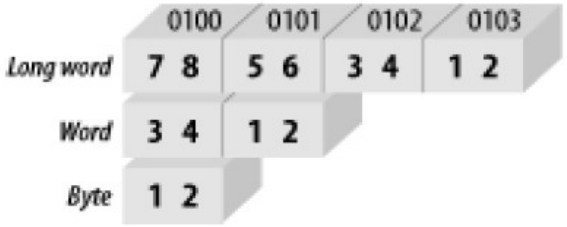



```
clear 0x1000
load r1,#5
```

```
xor r1,r1
store r1,0x1000
add r1,#5
```







Architektura procesora CISC

Cechy architektury CISC (Complex Instruction Set Computers):

- Duża liczba rozkazów (instrukcji),
- Niektóre rozkazy potrzebują dużej liczby cykli procesora do wykonania,
- Występowanie złożonych, specjalistycznych rozkazów,
- Duża liczba trybów adresowania,
- Do pamięci może się odwoływać bezpośrednio duża liczba rozkazów,
- Mniejsza od układów RISC częstotliwość taktowania procesora,
- Powolne działanie dekodera rozkazów, ze względu na dużą ich liczbę i skomplikowane adresowanie

Przykłady rodzin procesorów o architekturze CISC to:

- x86
- **M68000**
- PDP-11
- AMD

Architektura procesora RISC

Cechy architektury RISC (Reduced Instruction Set Computer):

- Zredukowana liczba rozkazów. Upraszcza to znacznie dekoder rozkazów.
- Redukcja trybów adresowania, dzięki czemu kody rozkazów są prostsze,
- Ograniczenie komunikacji pomiędzy pamięcią, a procesorem. Do przesyłania danych pomiędzy pamięcią, a rejestrami służą dedykowane instrukcje (load, store) .
- Zwiększenie liczby rejestrów (np. 32, 192, 256),
- Dzięki przetwarzaniu potokowemu (ang. pipelining) wszystkie rozkazy wykonują się w jednym cyklu maszynowym.

Przykłady rodzin mikroprocesorów o architekturze RISC:

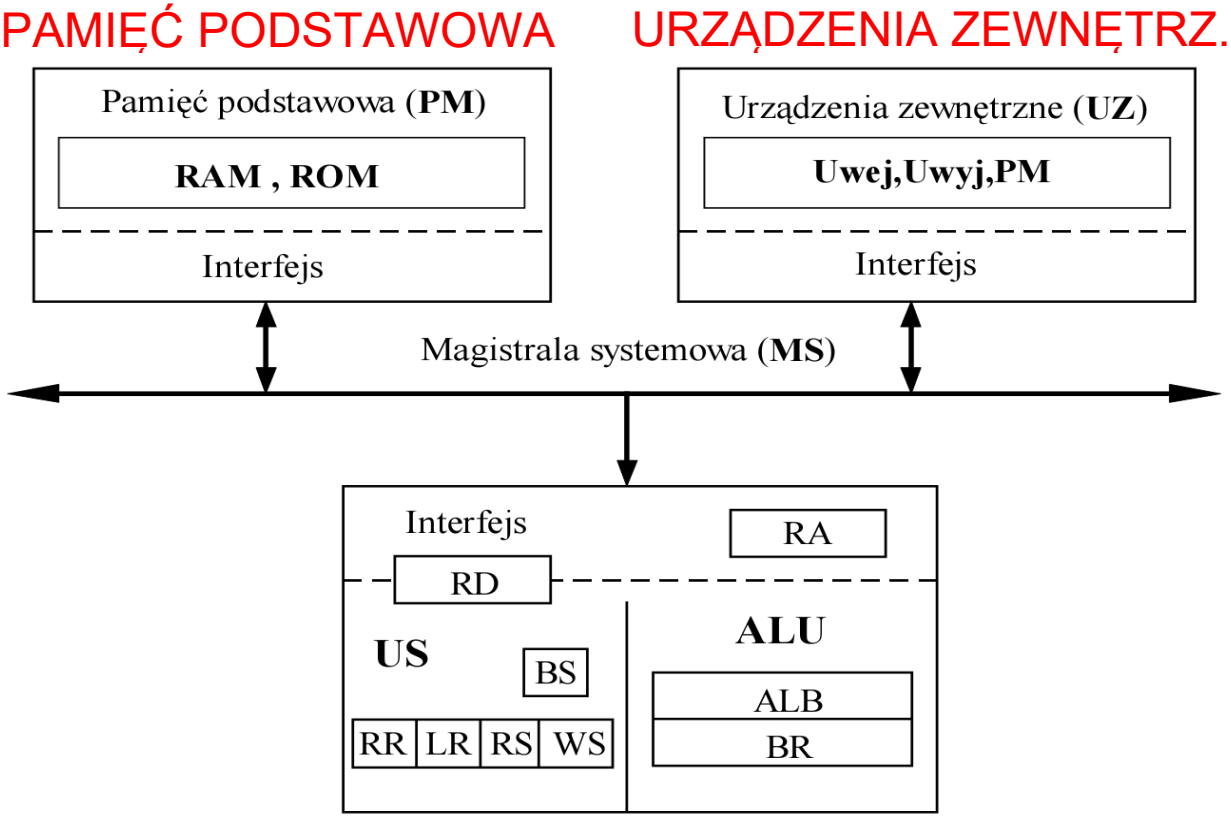
- IBM 801
- PowerPC
- MIPS
- Alpha
- **ARM**
- Motorola 88000
- ColdFire
- SPARC
- PA-RISC
- Atmel_AVR

Obecnie produkowane procesory Intela z punktu widzenia programisty są widziane jako CISC, ale ich rdzeń jest zgodny z RISC. Rozkazy CISC są rozbijane na mikrorozkazy (ang. microops), które są następnie wykonywane przez szybki blok wykonawczy zgodny z architekturą RISC.

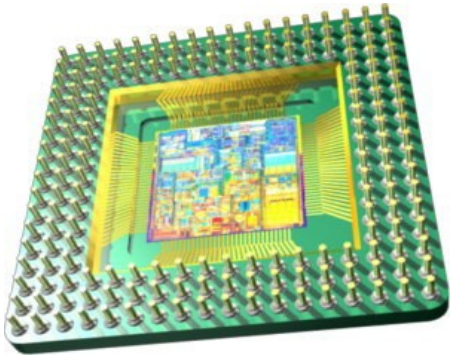
Architektura systemu komputerowego

Architektura polega na ścisłym podziale komputera na trzy podstawowe części:

- procesor,
- pamięć (zawierająca dane oraz program),
- urządzenia wejścia/wyjścia (I/O).



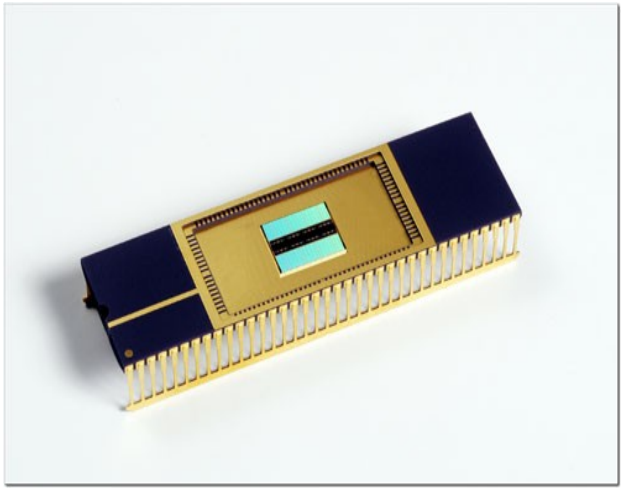
Magistrale komputera



Magistrala adresowa

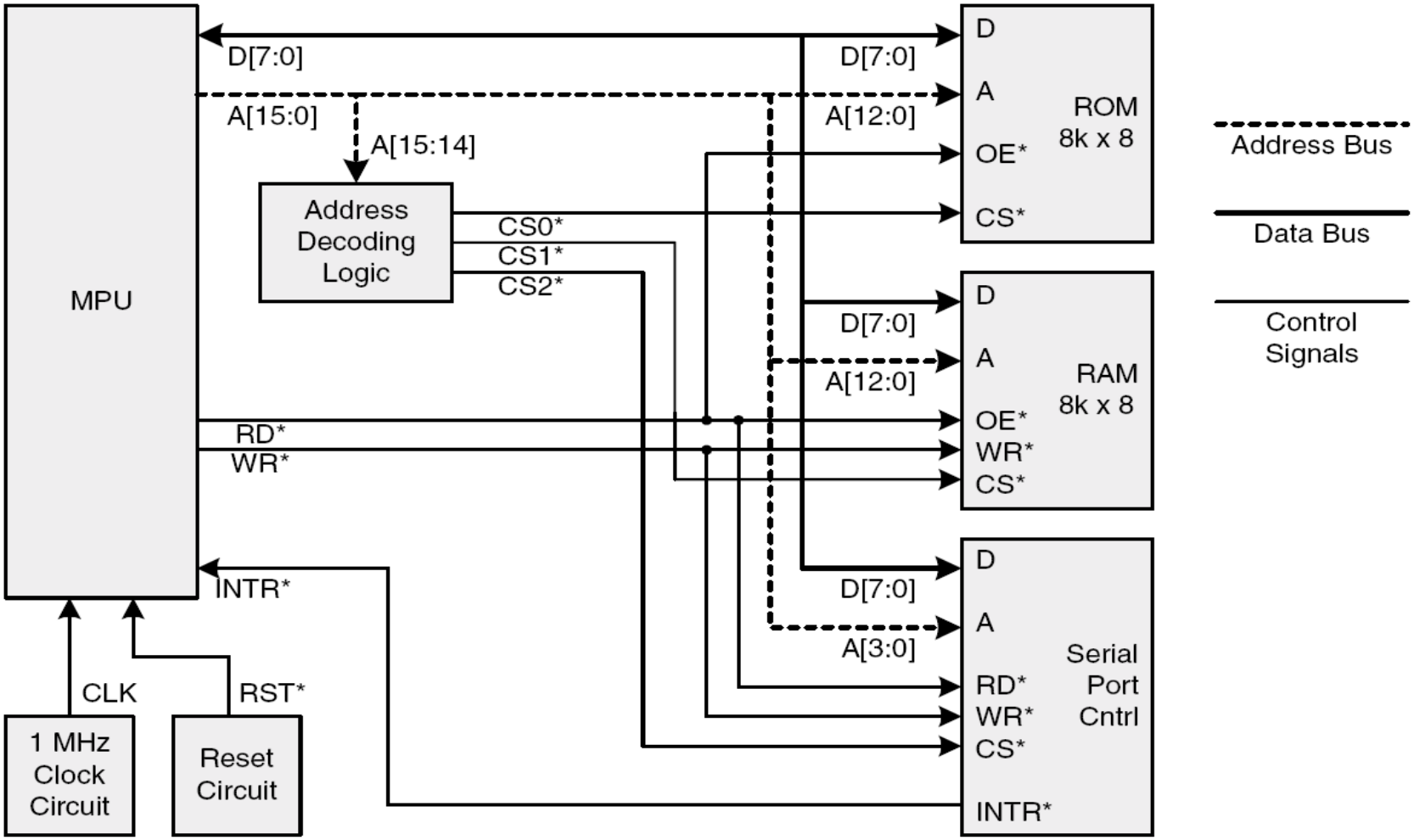
Magistrala danych

Magistrala sterująca



1. Rodzaj magistrali
2. Szerokość magistrali
3. Częstotliwość zegara – szybkość transmisji

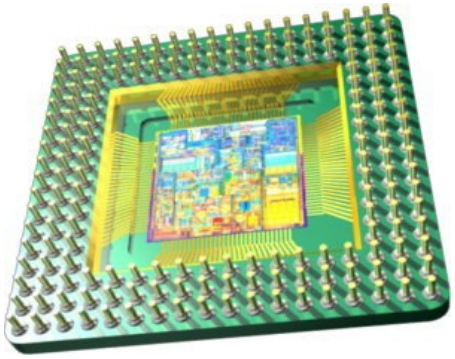
Przykładowy komputer 8-bitowy



Architektura von Neumanna

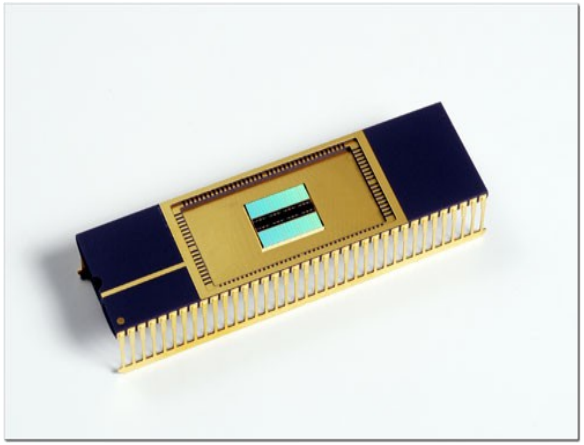
Cechy architektury von Neumanna:

- rozkazy i dane przechowywane są w tej samej pamięci,
- nie da się rozróżnić danych o rozkazów (instrukcji),
- dane nie maja przypisanego znaczenia,
- pamięć traktowana jest jako liniowa tablica komórek, które identyfikowane są przy pomocy dostarczanego przez procesor adresu,
- procesor ma dostęp do przestrzeni adresowej, dekodery adresowe zapewniają mapowanie pamięci na rzeczywiste układy.



Magistrala adresowa

Magistrala danych

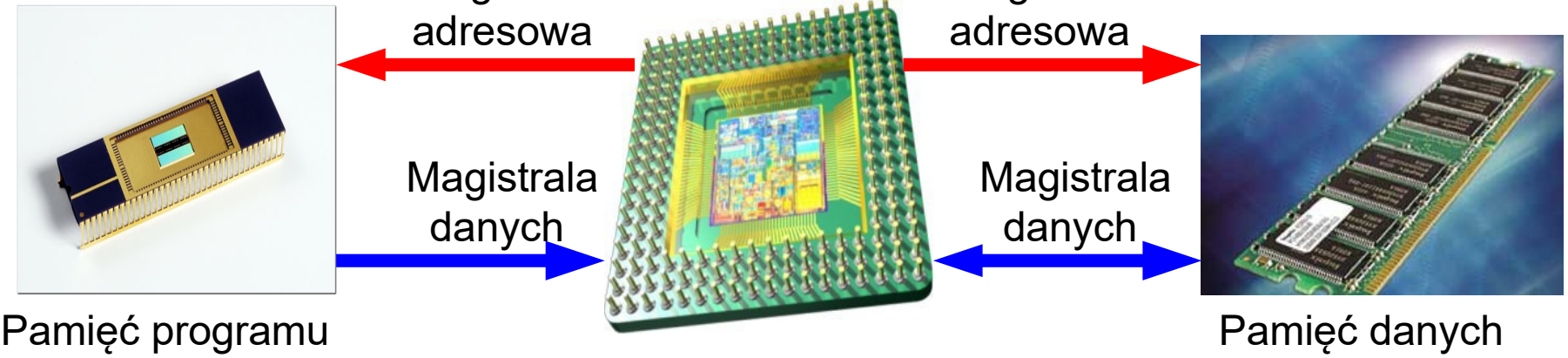


Architektura Harwardzka

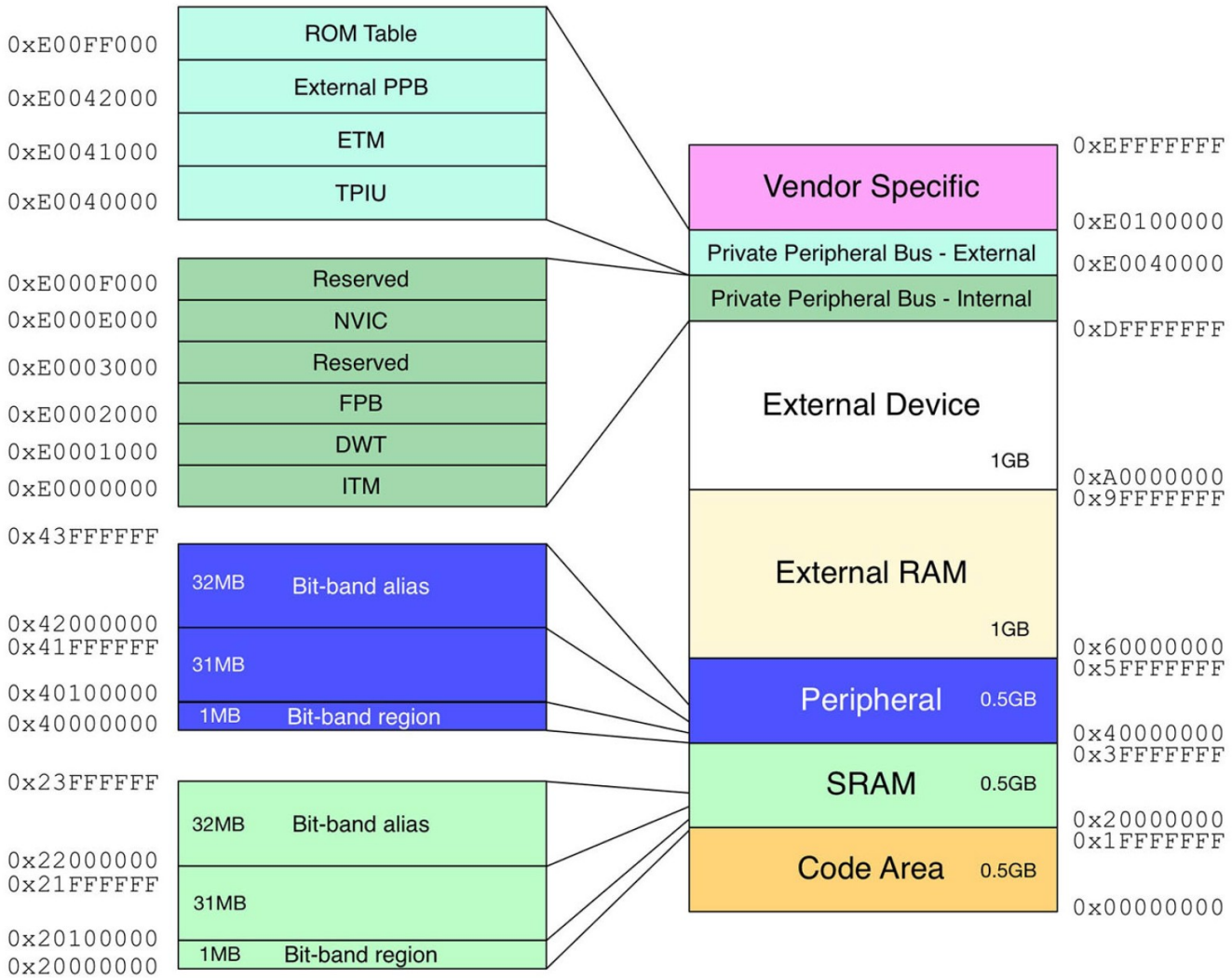
Prostsza (w stosunku do architektury Von Neumanna) budowa przekłada się na większą szybkość działania - dlatego ten typ architektury jest często wykorzystywany w procesorach sygnałowych oraz przy dostępie procesora do pamięci cache.

Cechy architektury Harwardzkiej:

- rozkazy i dane przechowywane są w oddzielnych pamięciach,
- organizacja pamięci może być różna (inne długości słowa danych i rozkazów),
- możliwość pracy równoległej – jednoczesny odczyt danych z pamięci programu oraz danych,
- stosowana w mikrokontrolerach jednoukładowych.



Memory Map for Cortex-M Microcontrollers



Rejestry

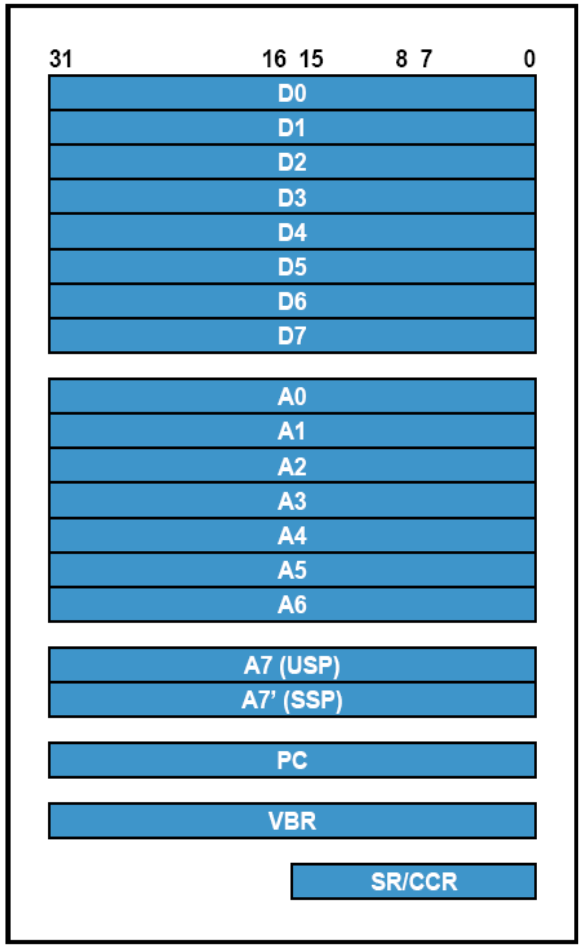
Rejestry procesora są realizowane jako komórki wewnętrznej pamięci procesora. Rejestry mają zazwyczaj niewielkie rozmiary (8/16/32/64/128 bitów). Rejestry służą do przechowywania chwilowych wyników obliczeń, adresów w pamięci komputera, konfiguracji urządzeń peryferyjnych itp.

Cecha rejestrów procesora:

- Najwyższy poziom w hierarchii pamięci (pamięć o najszybszym dostępie),
- Zaimplementowane jako przerzutniki bistabilne,
- Liczba rejestrów zależy od typu procesora (RISC/CISC).

Rejestry można podzielić na następujące grupy:

- Rejestry danych – służą do przechowywania danych i wyników, m.in. argumenty funkcji, wyniki obliczeń,
- Rejestry adresowe – wykorzystywane do operacji na adresach (wskaźnik stosu, licznik programu, rejestr segmentowy itp.),
- Rejestry ogólnego przeznaczenia – przechowują zarówno dane, jak i adresy,
- Rejestry zmiennoprzecinkowe – wykorzystywane do operacji na rejestrach zmiennoprzecinkowych (koprocesor FPU).



Rejestry mikrokontrolera Freescale/NXP

Kolejność bajtów w pamięci (1)

Bajt – najmniejsza adresowalna jednostka pamięci komputerowej

Endianess

Big-endian

...pod najmłodszym adresem umieszczony jest najstarszy bajt

podobnie jak w języku polskim, angielskim

Motorola, SPARC, ARM

middle-endian

liczby zmiennoprzecinkowe podwójnej precyzji

VAX and ARM

Little-endian

...pod najmłodszym adresem umieszczony jest najmłodszy bajt

podobnie jak w językach niemieckim, arabskich

Intel x86, 6502 VAX

Bi-Endian

ARM, PowerPC (za wyjątkiem PPC970/G5), DEC Alpha, MIPS, PA-RISC oraz IA64

Kolejność bajtów w pamięci (2)

Dane 8-bitowe, **Bajt 1, Bajt 2, Bajt 3,...**

7	0
0x0000.0000	Bajt 1
0x0000.0001	Bajt 2
0x0000.0002	Bajt 3
0x0000.0003	Bajt 4
0x0000.0004	Bajt 5

7	0
0x0000.0000	0x01
0x0000.0001	0x02
0x0000.0002	0x03
0x0000.0003	0x04
0x0000.0004	0x05

Kolejność bajtów w pamięci (2)

Dane 32-bitowe, Podwójne słowo (DW): **Bajt 4 … Bajt 1**

7	0
0x0000.0000	Bajt 4
0x0000.0001	Bajt 3
0x0000.0002	Bajt 2
0x0000.0003	Bajt 1
0x0000.0004	Bajt 8

Big-endian

7	0
0x0000.0000	Bajt 1
0x0000.0001	Bajt 2
0x0000.0002	Bajt 3
0x0000.0003	Bajt 4
0x0000.0004	Bajt 5

Little-endian

Kolejność bajtów w pamięci (2)

Dane 32-bitowe, Podwójne słowo (DW): **0x0403.0201**

	7 ... 0
0x0000.0000	0x04
0x0000.0001	0x03
0x0000.0002	0x02
0x0000.0003	0x01
0x0000.0004	0x08

Big-endian

	7 ... 0
0x0000.0000	0x01
0x0000.0001	0x02
0x0000.0002	0x03
0x0000.0003	0x04
0x0000.0004	0x05

Little-endian

Kolejność bajtów w pamięci (5)

Jak rozpoznać architekturę procesora oraz rozkład bajtów w pamięci?

```
#define LITTLE_ENDIAN   0
#define BIG_ENDIAN      1

int machineEndianness()
{
   long int nbr = 1;                              /* 32 bit = 0x0000.0001 */
   const char *ptr = (const char *) &nbr;         /* wskaźnik do ......?  */

   if (ptr[0] == 1)  /* Lowest address contains the least significant byte */
      return LITTLE_ENDIAN;

   else
      return BIG_ENDIAN;
}
```

Operacje na rejestrach

Operacje bitowe w języku C

volatile unsigned int * DataInMemory = 0x4800.0000;

volatile uint32_t * DataInMemory = 0x1000;

*DataInMemory = 0;

*DataInMemory = 0x12345678;

*DataInMemory = 16789U;

*DataInMemory = 0xFFFF.FFFFU;

Jak wyzerować pojedynczy bit ?

Jak ustawić pojedynczy bit ?

Operacje na rejestrach w języku C

```
volatile unsigned int* GPIOA_PUPDR = 0x4800.000C;
volatile uint32_t * GPIOB_PUPDR = 0x4800.040C;
#define GPIOC_PUPDR (volatile uint32_t *)0x4800.080C
    *GPIOA_PUPDR    = 0x1U;
    *GPIOA_PUPDR    = 7U;
    *GPIOA_PUPDR    = 010U;
    *GPIOA_PUPDR    = *GPIOA_PUPDR  |  0x2U;
    *GPIOA_PUPDR   |= 0x1U | 0x2U | 0x8U ;
    *GPIOA_PUPDR  &=  ~(0x2U | 0x4U);
    *GPIOA_PUPDR   ^=    (0x1U | 0x2U);
    *GPIOA_PUPDR   ^=     0x3U;
    If (*GPIOA_PUPDR & (0x1U | 0x4U)) == 0  {...}
    while (*GPIOA_PUPDR != 0x6U)   {...}
    do {...} while (*GPIOA_PUPDR & 0x4U)
```

Operacje na rejestrach w języku C (2)

```
#define PB0   0x1
#define PB1   0x2
#define PB2   1U<<2
#define PB3   1U<<3

volatile unsigned char* PORTA = (volatile unsigned char*) 0x4010.000A;

*PORTA   |= PB1 | PB2;
*PORTA &=  ~(PB1 | PB2);
*PORTA  ^=  (PB1 | PB2);
If (*PORTA & (PB1 | PB2)) == 0

enum {PB0=1U<<0, PB1=1U<<2, PB2=1U<<3, PB3=1U<<3};
```

Operacje na rejestrach w języku C (3)

```
volatile unsigned char* PORTA = (volatile unsigned char*) 0x4010.000A;

/* macro for bit-mask */
#define BIT(x)      (1u << (x))
*PORTA  |=  BIT(0);
*PORTA &=~BIT(1);
*PORTA  ^= BIT(2);

/* macro for setting and clearing bits */
#define SETBIT(P, B)        (P)   |=   BIT(B)
#define CLRBIT(P, B)        (P)  &= ~BIT(B)

SETBIT(*PORTA, 7);
CLRBIT(*PORTA, 2);
```

Konwersja 8, 16, 32-bity

```
int main(void) {
        uint8_t reg1=0x15, reg2=0x55;
        uint8_t  = reg3, reg4;
        uint16_t  tmp;
        /* concatenation operation */
        tmp = reg1;
        tmp = tmp<<8 | reg2;

        /* deconcatenation operation  */
        reg3 = tmp>>8;              /* be careful with signed numbers */
        reg4 = tmp & 0xFF;

}
```

Rejestry mapowane na strukturę

```
typedef volatile unsigned int AT91_REG;       // Hardware register definition
typedef struct _AT91S_PIO {
        AT91_REG        PIO_PER;                // PIO Enable Register, 32-bit register
        AT91_REG        PIO_PDR;                // PIO Disable Register
        AT91_REG        PIO_PSR;                // PIO Status Register
        AT91_REG        Reserved1[1];           //
        AT91_REG        PIO_IFER;               // Input Filter Enable Register
        AT91_REG        PIO_IFDR;               // Input Filter Disable Register
        AT91_REG        PIO_IFSR;               // Input Filter Status Register
        AT91_REG        Reserved2[1];           //
} AT91S_PIO, *AT91PS_PIO;
        /* registers for paraller port of ARM processor I/O PIOA...PIOE */
#define AT91C_BASE_PIOA     (AT91PS_PIO)    0xFFFFF200U     // (PIOA) Base Address
        /* mask for zero bit of port PA */
#define AT91C_PIO_PA0       (1 <<  0)       // Pin Controlled by PA0
```

How to set 0 and 19th bith of register PIO_PER ?

```
AT91C_BASE_PIOA->PIO_PER |= AT91C_PIO_PA0 | AT91C_PIO_PA19;
```

Rejestry mapowane na strukturę – pola bitowe

```
Struct Port_4bit {
unsigned Bit_0          :     1;
unsigned Bit_1          :     1;
unsigned Bit_2          :     1;
unsigned Bit_3          :     1;
unsigned Bit_Filler     :     4;
};
#define PORTC (*(Port_4bit*)0x4010.0002U)
int i = PORTC.Bit_0;        /* read data */
PORTC.Bit_2 = 1;            /* write data */

Port_4bit* PortTC = (Port_4bit*) 0x4010.000FU;
int i = PortTC->Bit_0;
PortTC->Bit_0 = 1;
```

- Bit-fields allows to 'pack' data – usage of single bits, e.g. bit flags
- Increase of code complexity required for operations on registers
- Bit-fields can be mapped in different ways in memory according different compilers and processors architectures
- Cannot use ***offsetof*** macro to calculate data offset in structure
- Cannot use ***sizeof*** macro to calculate size of data
- Tables cannot use bit-fields

Struktury, Unie – dwie różne funkcje

```
extern volatile union {
        struct {
                unsigned EID16      :1;
                unsigned EID17      :1;
                unsigned            :1;
                unsigned EXIDE      :1;
                unsigned            :1;
                unsigned SID0       :1;
                unsigned SID1       :1;
                unsigned SID2       :1;
        };
        struct {
                unsigned            :3;
                unsigned EXIDEN     :1;
        };
} RXF3SIDLbits_;
```

Structures have the same address:

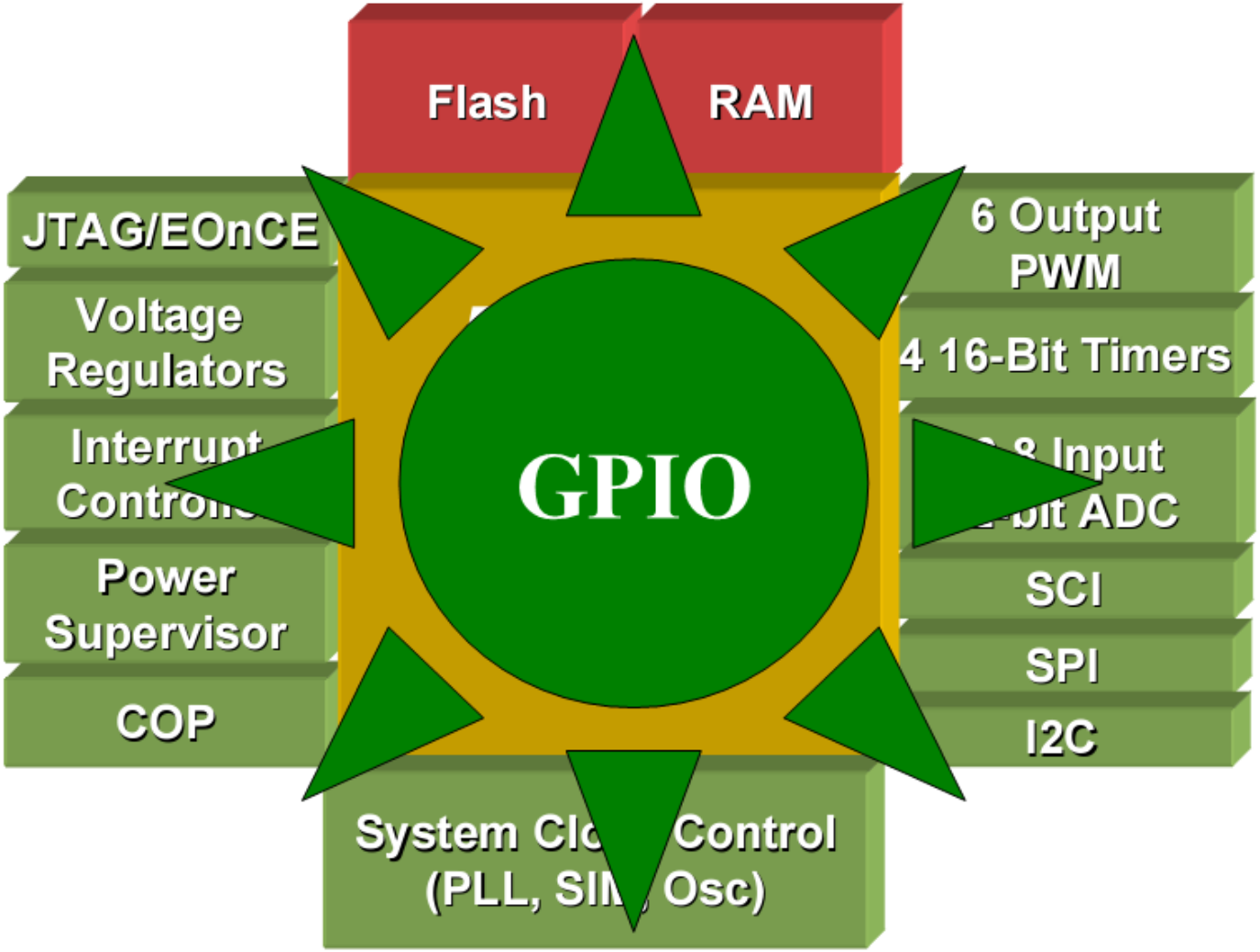
```
#define RXF3SIDLbits
    (*(Port_RXF3SIDLbits_*)0x4010.0000)
```

Access to data mapped into structure:

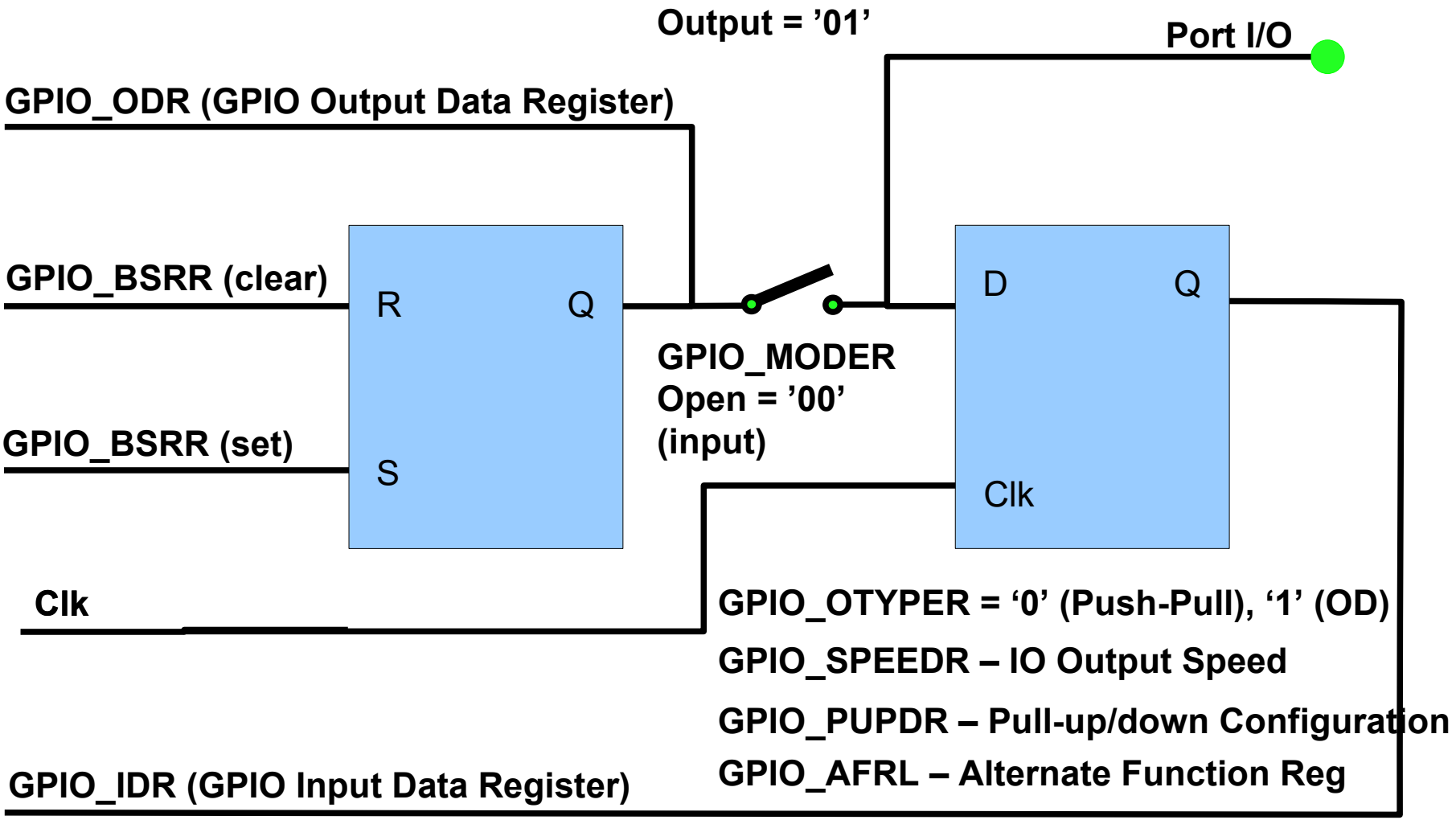
```
/* data in first structure */
      RXF3SIDLbits.EID16 = 1;

/* data in second structure */
      RXF3SIDLbits.EXIDEN = 0;
```

Sterownik portów wejścia-wyjścia (Input-Output)

Porty Wejścia-Wyjścia (IO/GPIO)



Uproszczony schemat blokowy sterownika I/O



Sterownik portów wejścia-wyjścia (GPIO) (1)

8 General-purpose I/Os (GPIO)

8.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR) and a 32-bit set/reset register (GPIOx_BSRR). In addition all GPIOs have a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

8.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bitwise write access to GPIOx_ODR
- Locking mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

Sterownik portów wejścia-wyjścia (2)

Każdy z portów IO może pracować w jednym z następujących trybów:

Wejście (Input)

- **Wejście pływające (floating input)**
- **Wejście z podciąganiem** do VCC **(input** with VCC **pull-up)**
- **Wejście z podciąganiem** do GND (**input** with GND **pull-down)**
- **Wejście analogowe (analogue input)**

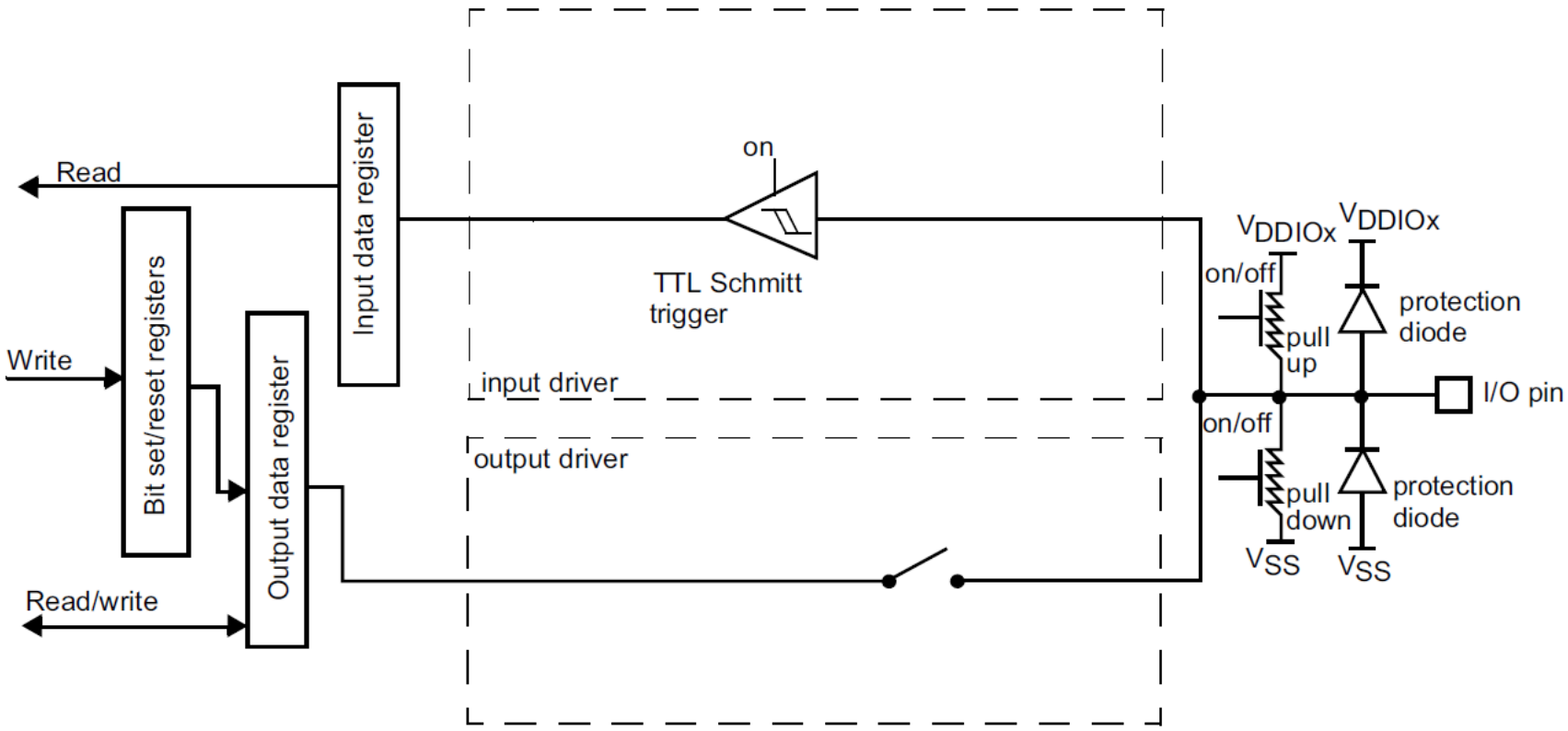
Wyjście (Output)

- **Otwarty Dren (Open drain output** with optional pull-up or pull-down function)
- **Wyjście push-pull** (with optional pull-up or pull-down function)

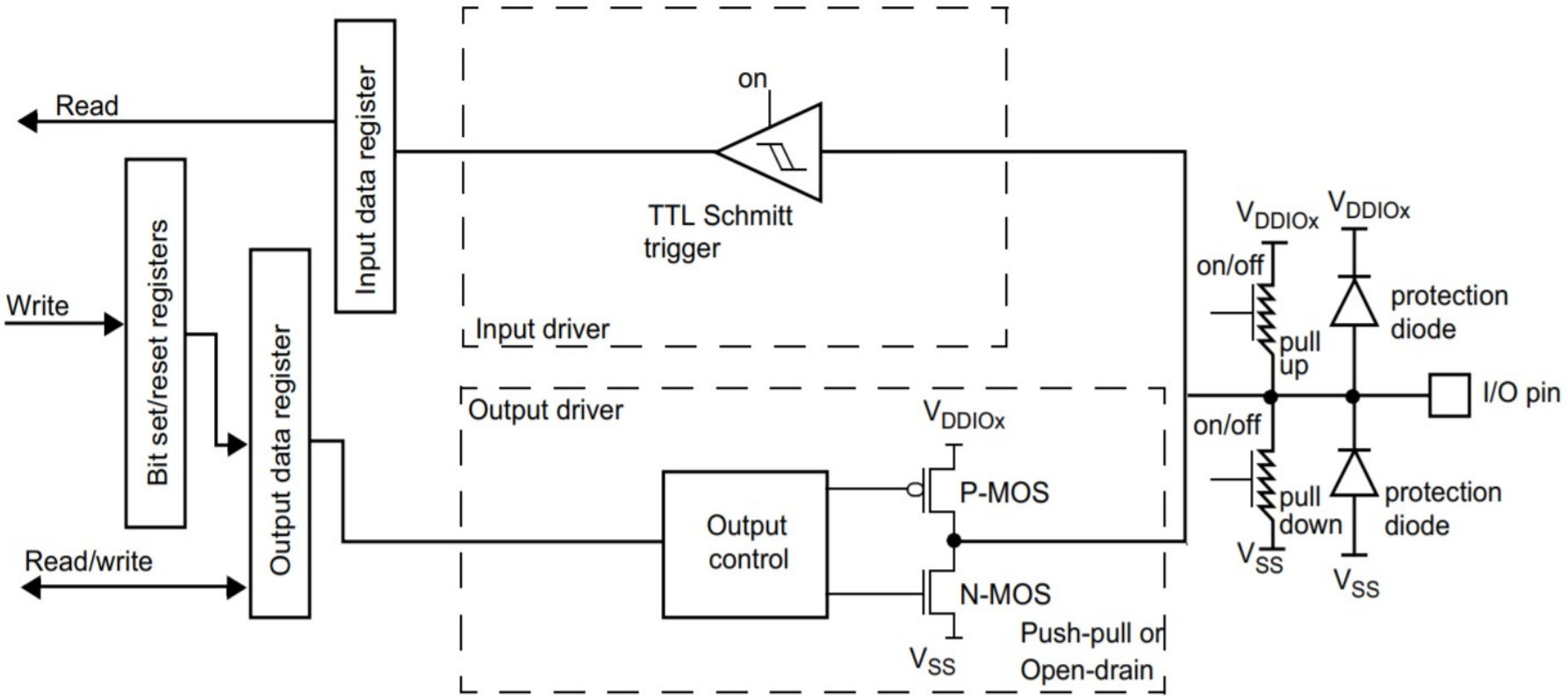
Funkcje alternatywne

- Alternative open drain function (with optional pull-up or pull-down function)
- Alternative function in push-pull mode (with optional pull-up or pull-down function)

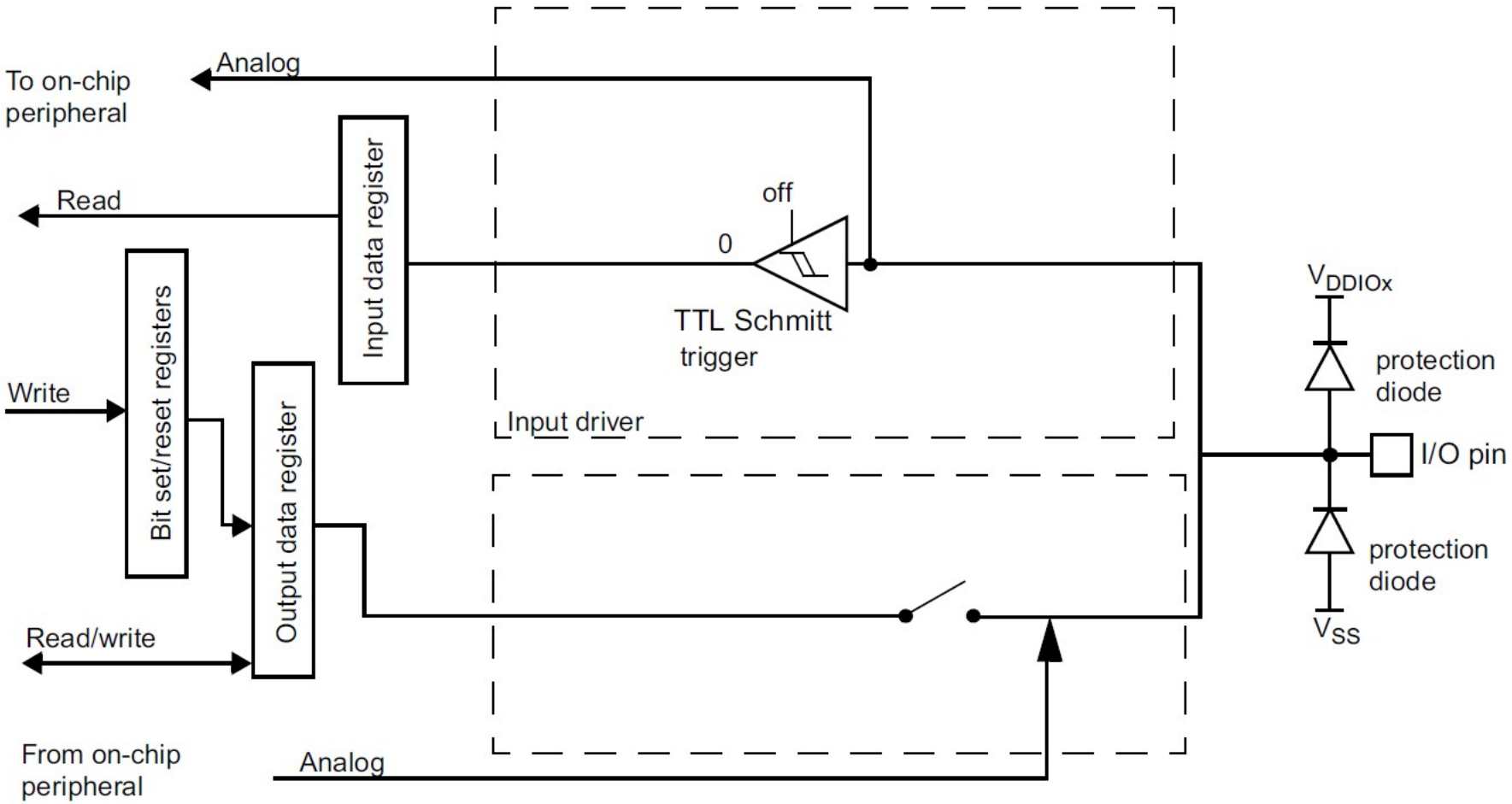
Port wejściowy



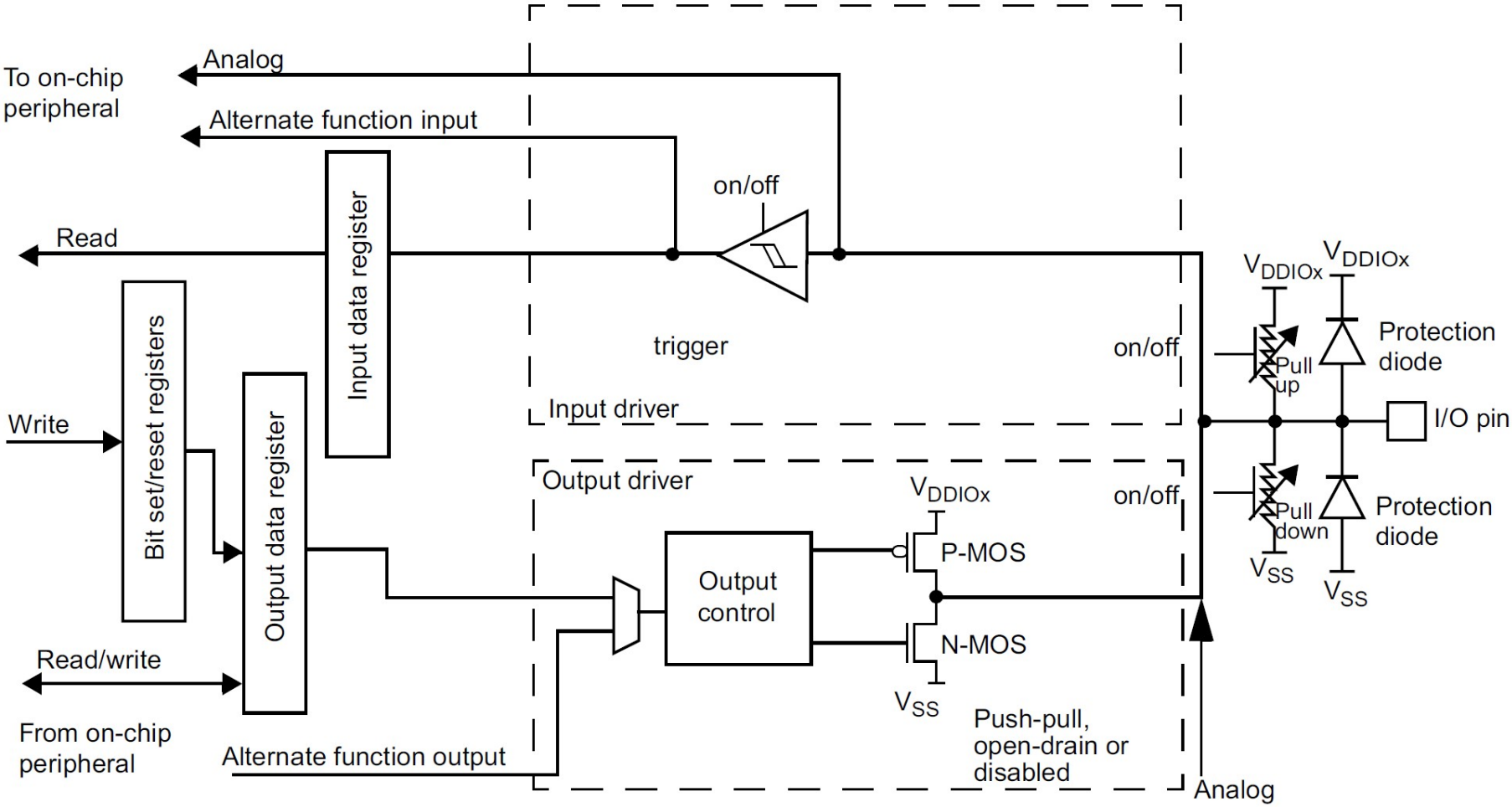
Port wyjściowy



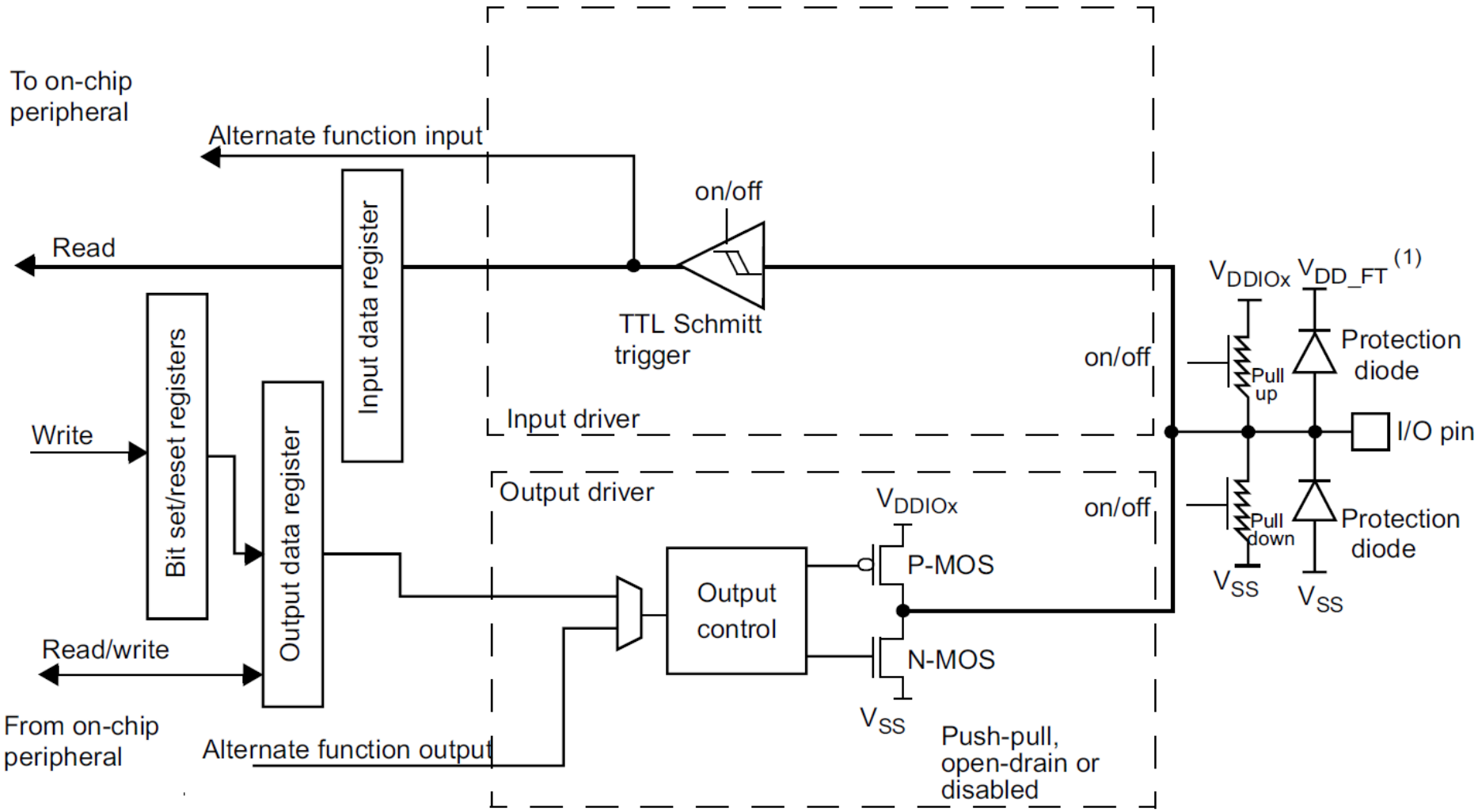
Porty analogowe



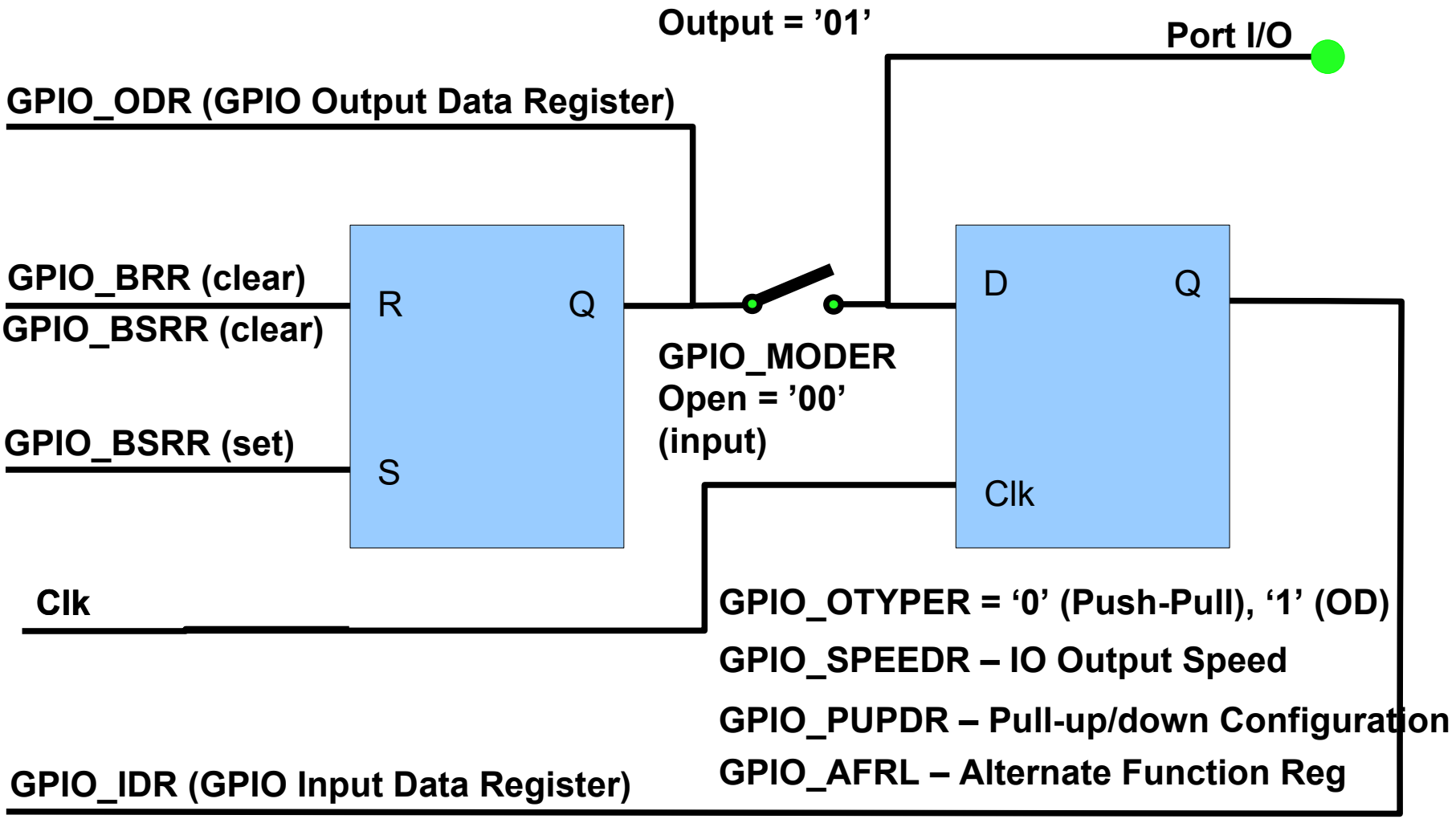
Połączenie portu IO z procesorem ARM



Porty tolerujące napięcia 5 V

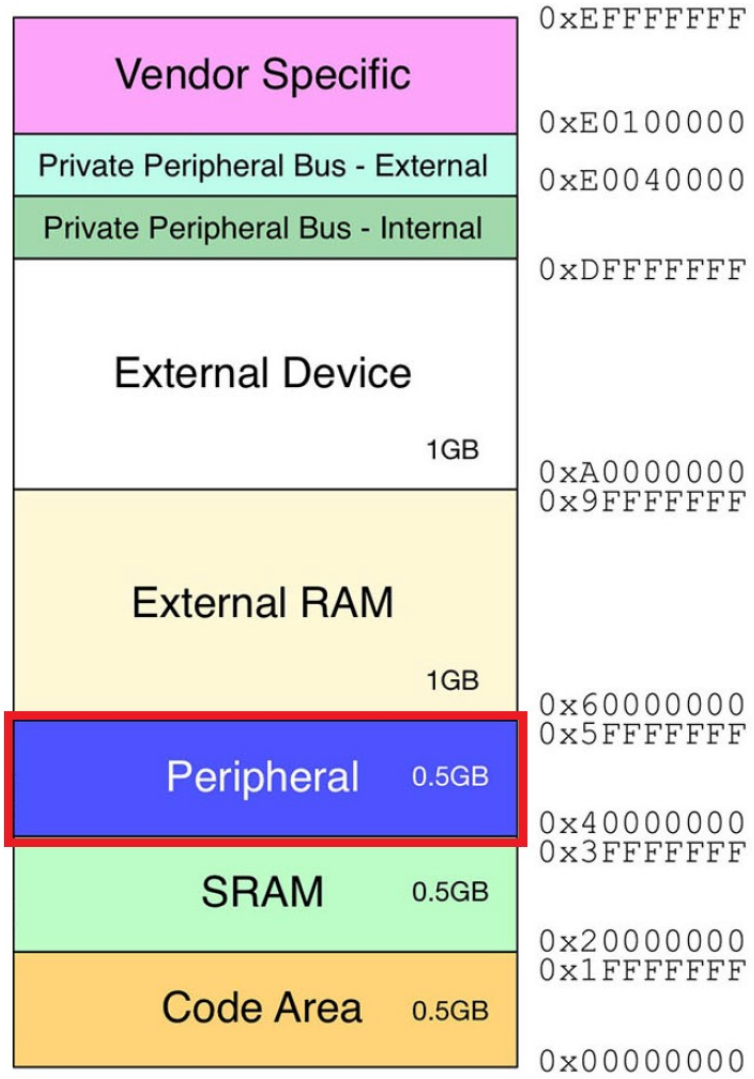


Uproszczony schemat blokowy sterownika I/O



Rejestry sterujące portem IO (1)

- Procesor jest wyposażony w 8 (porty A-H) 16-bitowych portów IO
- Obsługuje do 128 sygnałów IO
- Każdy port jest kontrolowany za pomocą 12 32-bitowych rejestrów umieszczonych w 1 kB pamięci
- Wszystkie rejestry są dostępne w trybie odczytu/zapisu, z wyjątkiem:
  - IDR – tylko do odczytu
  - BSRR (BSR/BRR) – tylko zapis (sterowanie bitowe)



Rejestry sterujące portem IO (2)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOA_MODER**	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**GPIOx_OTYPER** (where x = A..I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOA_OSPEEDR**	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR**	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = A..I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Rejestry sterujące portem IO (3)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	**GPIOx_ODR** (where x = A..I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = A..I)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR** (where x = A..I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL** (where x = A..I)	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH** (where x = A..I)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**GPIOx_BRR** (where x = A..I)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**GPIOx_ASCR** (where x = A..H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASC15	ASC14	ASC13	ASC12	ASC11	ASC10	ASC9	ASC8	ASC7	ASC6	ASC5	ASC4	ASC3	ASC2	ASC1	ASC0
	Reset value	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Rejestry sterujące portem IO - podsumowanie

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

Adresy bazowe portów A-H

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB2	0x4800 1C00 - 0x4800 1FFF	1 KB	GPIOH	*Section 8.4.13: GPIO register map*
	0x4800 1800 - 0x4800 1BFF	1 KB	GPIOG	*Section 8.4.13: GPIO register map*
	0x4800 1400 - 0x4800 17FF	1 KB	GPIOF	*Section 8.4.13: GPIO register map*
	0x4800 1000 - 0x4800 13FF	1 KB	GPIOE	*Section 8.4.13: GPIO register map*
	0x4800 0C00 - 0x4800 0FFF	1 KB	GPIOD	*Section 8.4.13: GPIO register map*
	0x4800 0800 - 0x4800 0BFF	1 KB	GPIOC	*Section 8.4.13: GPIO register map*
	0x4800 0400 - 0x4800 07FF	1 KB	GPIOB	*Section 8.4.13: GPIO register map*
	0x4800 0000 - 0x4800 03FF	1 KB	GPIOA	*Section 8.4.13: GPIO register map*
	0x4002 4400 - 0x47FF FFFF	~127 MB	Reserved	-

Sygnał zegara dla portów IO

6.4.17 AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x4C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: *When the peripheral clock is not active, the peripheral registers read or write access is not supported.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG EN	HASHEN	AESEN (1)
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DCMIEN	ADCEN	OTGFSEN	Res.	Res.	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

1. Available on STM32L42xxx, STM32L44xxx and STM32L46xxx devices only.

Sygnał zegara dla portów GPIO (2)

Table 34. RCC register map and reset values (continued)

Off-set	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4C	RCC_AHB2 ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEN	HASHEN	AESEN	Res.	DCMIEN	ADCEN	OTGFSEN	Res.	Res.	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value														0	0	0		0	0	0				0	0	0	0	0	0	0	0	0
0x50	RCC_AHB3 ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QSPIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCEN
	Reset value																								0								0

Sterowanie zasilaniem portów IO

5.4.2 Power control register 2 (PWR_CR2)

Address offset: 0x04

Reset value: 0x0000 0000. This register is reset when exiting the Standby mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	USV	IOSV	Res.	PVME4	PVME3	PVME2	PVME1	PLS[2:0]			PVDE
					rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **USV**: V_{DDUSB} USB supply valid
This bit is used to validate the V_{DDUSB} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the USB OTG_FS peripheral. If V_{DDUSB} is not always present in the application, the PVM can be used to determine whether this supply is ready or not.
0: V_{DDUSB} is not present. Logical and electrical isolation is applied to ignore this supply.
1: V_{DDUSB} is valid.

Bit 9 **IOSV**: V_{DDIO2} Independent I/Os supply valid
This bit is used to validate the V_{DDIO2} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PG[15:2]. If V_{DDIO2} is not always present in the application, the PVM can be used to determine whether this supply is ready or not.
0: V_{DDIO2} is not present. Logical and electrical isolation is applied to ignore this supply.
1: V_{DDIO2} is valid.

Sterowanie zasilaniem portów IO (2)

Podczas pierwszych zajęć warto użyć funkcji HAL:

- __STATIC_INLINE void LL_PWR_EnableVddIO2 (void)

W kolejnym ćwiczeniu funkcję należy napisać samodzielnie

- Potrzebny będzie adres bazowy (Base Address) dla urządzenia PWR

0x4000 7000	0x4000 73FF	1 KB	PWR	*Section 5.4.26: PWR register map and reset value table*

5.4.2 Power control register 2 (PWR_CR2)

Address offset: 0x04

Reset value: 0x0000 0000. This register is reset when exiting the Standby mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	USV	IOSV	Res.	PVME4	PVME3	PVME2	PVME1	PLS[2:0]			PVDE
					rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Sterowanie zegarem urządzeń peryferyjnych

6.4.19 APB1 peripheral clock enable register 1 (RCC_APB1ENR1)

Address: 0x58

Reset value: 0x0000 0400 (for STM32L496xx/4A6xx devices)
0x0000 0000 (for STM32L475xx/476xx/486xx devices)

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM1 EN	OPAMP EN	DAC1 EN	PWR EN	Res.	CAN2 EN	CAN1 EN	CRSEN	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN(1)	USART3 EN	USART2 EN	Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Res.	Res.	WWD GEN	RTCA PBEN	LCD EN	Res.	Res.	Res.	TIM7 EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2 EN
rw	rw			rs	rw	rw				rw	rw	rw	rw	rw	rw

1. Available on STM32L45xxx and STM32L46xxx devices only.

Bit 28 **PWREN**: Power interface clock enable
Set and cleared by software.
0: Power interface clock disabled
1: Power interface clock enabled

Adresy Bazowe

- PWR – Base Address

APB1	0x4000 7800 - 0x4000 7BFF	1 KB	OPAMP	*Section 23.5.7: OPAMP register map*
	0x4000 7400 - 0x4000 77FF	1 KB	DAC1	*Section 19.7.21: DAC register map*
	0x4000 7000 - 0x4000 73FF	1 KB	PWR	*Section 5.4.26: PWR register map and reset value table*

- RCC – Base Address

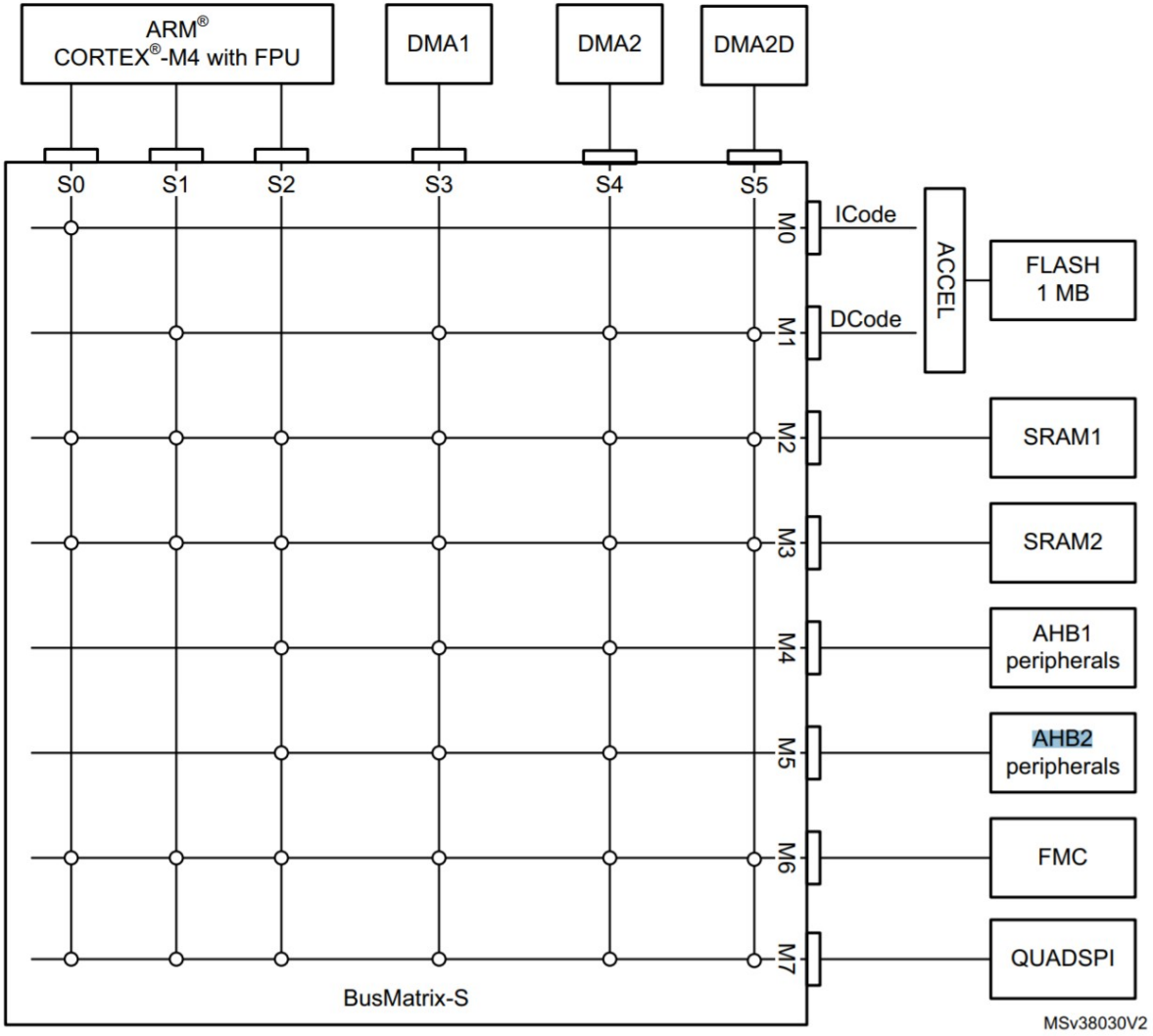
AHB1	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 KB	FLASH registers	*Section 3.7.17: FLASH register map*
	0x4002 1400 - 0x4002 1FFF	3 KB	Reserved	-
	0x4002 1000 - 0x4002 13FF	1 KB	RCC	*Section 6.4.33: RCC register map*
	0x4002 0800 - 0x4002 0FFF	2 KB	Reserved	-

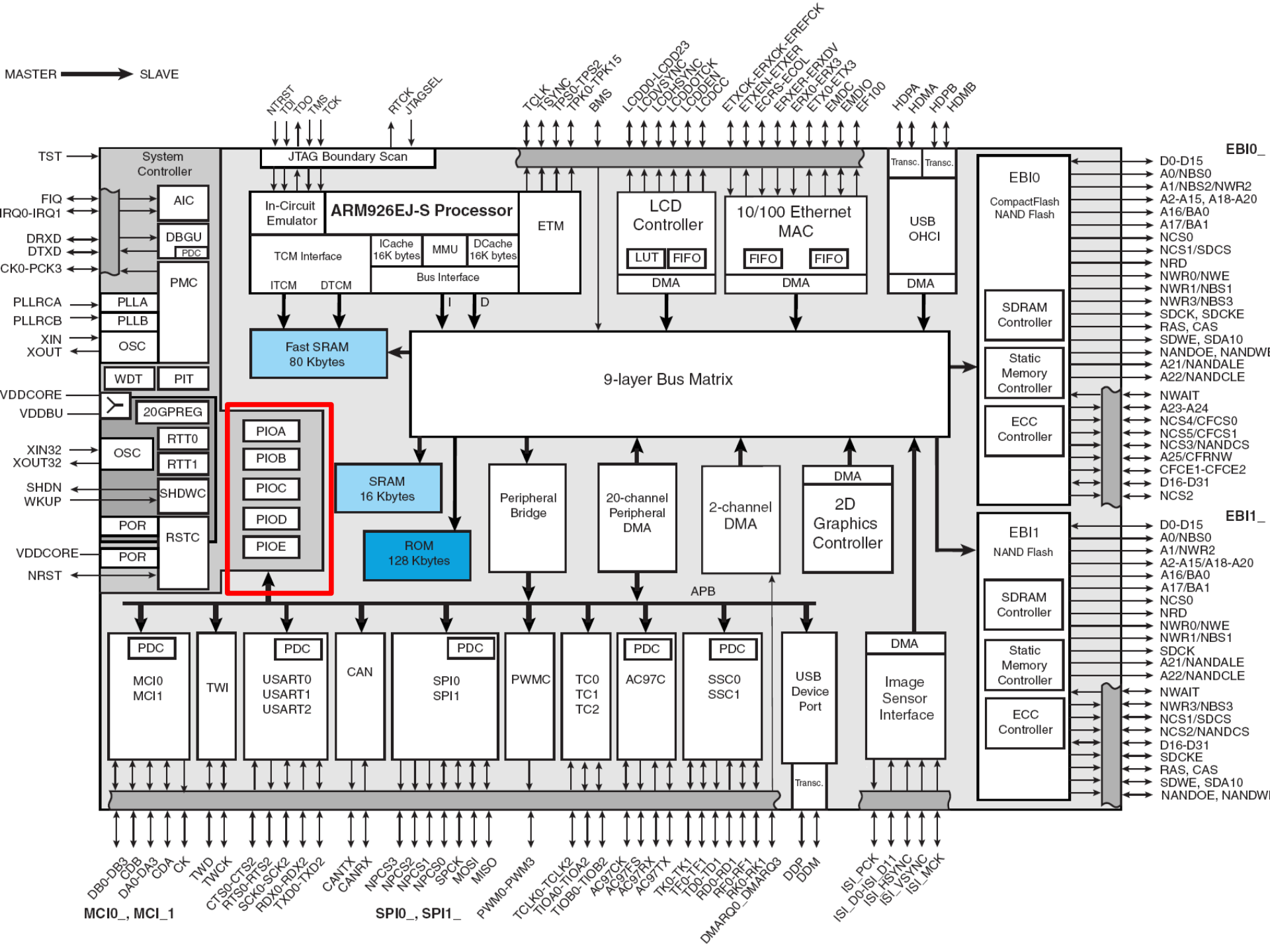
Architektura mikrokontrolera STM32L4

The main system consists of 32-bit multilayer AHB bus matrix that interconnects:

- **Up to six masters:**
  - – Cortex®-M4 with FPU core I-bus
  - – Cortex®-M4 with FPU core D-bus
  - – Cortex®-M4 with FPU core S-bus
  - – DMA1
  - – DMA2
  - – DMA2D (only for STM32L496xx/4A6xx devices)
- **Up to eight slaves:**
  - – Internal Flash memory on the ICode bus
  - – Internal Flash memory on DCode bus
  - – Internal SRAM1 (96 KB for STM32L475xx/476xx/486xx devices, 256 KB for STM32L496xx/4A6xx devices)
  - – Internal SRAM2 (32 KB for STM32L475xx/476xx/486xx devices, 64 KB for STM32L496xx/4A6xx devices)
  - – AHB1 peripherals including AHB to APB bridges and APB peripherals (connected to APB1 and APB2)
  - – **AHB2 peripherals**
  - – Flexible Memory Controller (FMC)
  - – Quad SPI memory interface (QUADSPI)

Architektura mikrokontrolera STM32L4 (1)




MASTER
SLAVE
NTRST
TDI
TDO
TMS
TCK
RTCK
JTAGSEL
TCLK
TSYNC
TPS0-TPS2
TPK0-TPK15
BMS
LCDD0-LCDD23
LCDVSYNC
LCDHSYNC
LCDDOTCK
LCDDEN
LCDCC
ETXCK-ERXCK-EREFCK
ETXEN-ETXER
ECRS-ECOL
ERXER-ERXDV
ERX0-ERX3
ETX0-ETX3
EMDC
EMDIO
EF100
HDPA
HDMA
HDPB
HDMB
TST
System Controller
JTAG Boundary Scan
FIQ
IRQ0-IRQ1
AIC
DRXD
DTXD
DBGU
PDC
CK0-PCK3
PMC
PLLRCA
PLLRCB
PLLA
PLLB
XIN
XOUT
OSC
WDT
PIT
VDDCORE
VDDBU
20GPREG
XIN32
XOUT32
RTT0
RTT1
SHDN
WKUP
SHDWC
POR
RSTC
NRST
In-Circuit Emulator
ARM926EJ-S Processor
TCM Interface
ICache 16K bytes
MMU
DCache 16K bytes
Bus Interface
ITCM
DTCM
ETM
I
D
LCD Controller
LUT
FIFO
DMA
10/100 Ethernet MAC
FIFO
FIFO
DMA
Transc.
Transc.
USB OHCI
DMA
Fast SRAM 80 Kbytes
9-layer Bus Matrix
PIOA
PIOB
PIOC
PIOD
PIOE
SRAM 16 Kbytes
ROM 128 Kbytes
Peripheral Bridge
20-channel Peripheral DMA
2-channel DMA
DMA
2D Graphics Controller
APB
EBI0
CompactFlash NAND Flash
SDRAM Controller
Static Memory Controller
ECC Controller
EBI0_
D0-D15
A0/NBS0
A1/NBS2/NWR2
A2-A15, A18-A20
A16/BA0
A17/BA1
NCS0
NCS1/SDCS
NRD
NWR0/NWE
NWR1/NBS1
NWR3/NBS3
SDCK, SDCKE
RAS, CAS
SDWE, SDA10
NANDOE, NANDWE
A21/NANDALE
A22/NANDCLE
NWAIT
A23-A24
NCS4/CFCS0
NCS5/CFCS1
NCS3/NANDCS
A25/CFRNW
CFCE1-CFCE2
D16-D31
NCS2
EBI1
NAND Flash
SDRAM Controller
Static Memory Controller
ECC Controller
EBI1_
D0-D15
A0/NBS0
A1/NWR2
A2-A15/A18-A20
A16/BA0
A17/BA1
NCS0
NRD
NWR0/NWE
NWR1/NBS1
SDCK
A21/NANDALE
A22/NANDCLE
NWAIT
NWR3/NBS3
NCS1/SDCS
NCS2/NANDCS
D16-D31
SDCKE
RAS, CAS
SDWE, SDA10
NANDOE, NANDWE
PDC
MCI0
MCI1
TWI
PDC
USART0
USART1
USART2
CAN
PDC
SPI0
SPI1
PWMC
TC0
TC1
TC2
PDC
AC97C
PDC
SSC0
SSC1
USB Device Port
Transc.
DMA
Image Sensor Interface
DB0-DB3
CDB
DA0-DA3
CDA
CK
TWD
TWCK
CTS0-CTS2
RTS0-RTS2
SCK0-SCK2
RDX0-RDX2
TXD0-TXD2
CANTX
CANRX
NPCS3
NPCS2
NPCS1
NPCS0
SPCK
MOSI
MISO
PWM0-PWM3
TCLK0-TCLK2
TIOA0-TIOA2
TIOB0-TIOB2
AC97CK
AC97FS
AC97RX
AC97TX
TK0-TK1
TF0-TF1
TD0-TD1
RD0-RD1
RF0-RF1
RK0-RK1
DMARQ0_DMARQ3
DDP
DDM
ISI_PCK
ISI_D0-ISI_D11
ISI_HSYNC
ISI_VSYNC
ISI_MCK
MCI0_, MCI_1
SPI0_, SPI1_

Documentation for AT91SAM9263 Microcontroller

Features

- Incorporates the ARM926EJ-S™ ARM® Thumb® Processor
  - DSP Instruction Extensions, Jazelle® Technology for Java® Acceleration
  - 16 Kbyte Data Cache, 16 Kbyte Instruction Cache, Write Buffer
  - 220 MIPS at 200 MHz
  - Memory Management Unit
  - EmbeddedICE™, Debug Communication Channel Support
  - Mid-level Implementation Embedded Trace Macrocell™
- Bus Matrix
  - Nine 32-bit-layer Matrix, Allowing a Total of 28.8 Gbps of On-chip Bus Bandwidth
  - Boot Mode Select Option, Remap Command
- Embedded Memories
  - One 128 Kbyte Internal ROM, Single-cycle Access at Maximum Bus Matrix Speed
  - One 80 Kbyte Internal SRAM, Single-cycle Access at Maximum Processor or Bus Matrix Speed
  - One 16 Kbyte Internal SRAM, Single-cycle Access at Maximum Bus Matrix Speed
- Dual External Bus Interface (EBI0 and EBI1)
  - EBI0 Supports SDRAM, Static Memory, ECC-enabled NAND Flash and CompactFlash®
  - EBI1 Supports SDRAM, Static Memory and ECC-enabled NAND Flash
- DMA Controller (DMAC)
  - Acts as one Bus Matrix Master
  - Embeds 2 Unidirectional Channels with Programmable Priority, Address Generation, Channel Buffering and Control
- Twenty Peripheral DMA Controller Channels (PDC)
- LCD Controller
  - Supports Passive or Active Displays
  - Up to 24 bits per Pixel in TFT Mode, Up to 16 bits per Pixel in STN Color Mode
  - Up to 16M Colors in TFT Mode, Resolution Up to 2048x2048, Supports Virtual Screen Buffers

ATMEL®

AT91 ARM Thumb Microcontrollers

AT91SAM9263

Preliminary

Documentation for AT91SAM9263 – I/O Ports

31. Parallel Input/Output Controller (PIO)

31.1 Overview

The Parallel Input/Output Controller (PIO) manages up to 32 fully programmable input/output lines. Each I/O line may be dedicated as a general-purpose I/O or be assigned to a function of an embedded peripheral. This assures effective optimization of the pins of a product.

Each I/O line is associated with a bit number in all of the 32-bit registers of the 32-bit wide User Interface.

Each I/O line of the PIO Controller features:

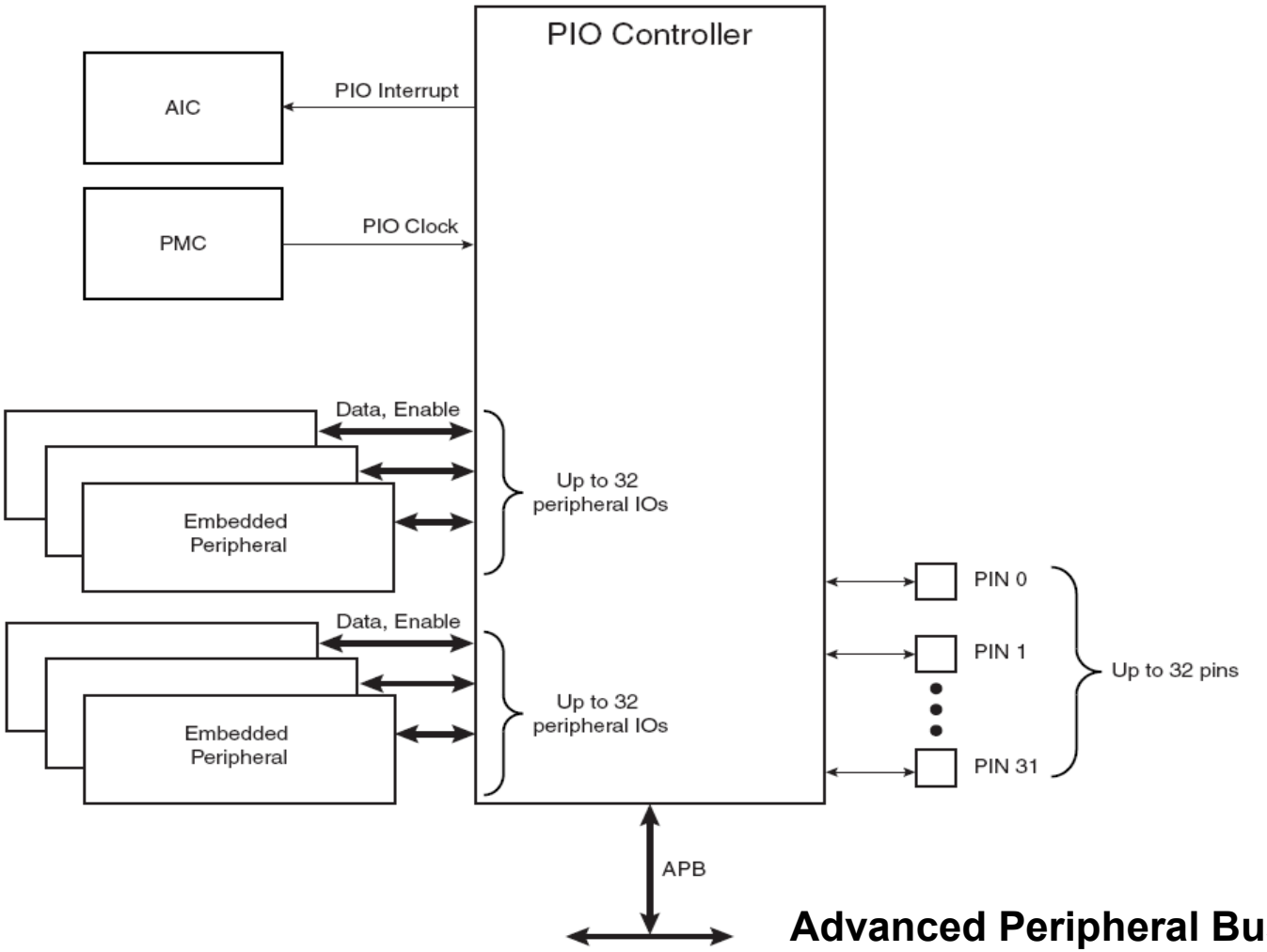
- An input change interrupt enabling level change detection on any I/O line.
- A glitch filter providing rejection of pulses lower than one-half of clock cycle.
- Multi-drive capability similar to an open drain I/O line.
- Control of the the pull-up of the I/O line.
- Input visibility and output control.

The PIO Controller also features a synchronous output providing up to 32 bits of data output in a single write operation.

Źródło: ATMEL, doc6249.pdf, strona 425

Block Diagram of 32-bits I/O Port

Figure 31-1. Block Diagram



Power Consumption vs Clock Signal

31.3.3 Power Management

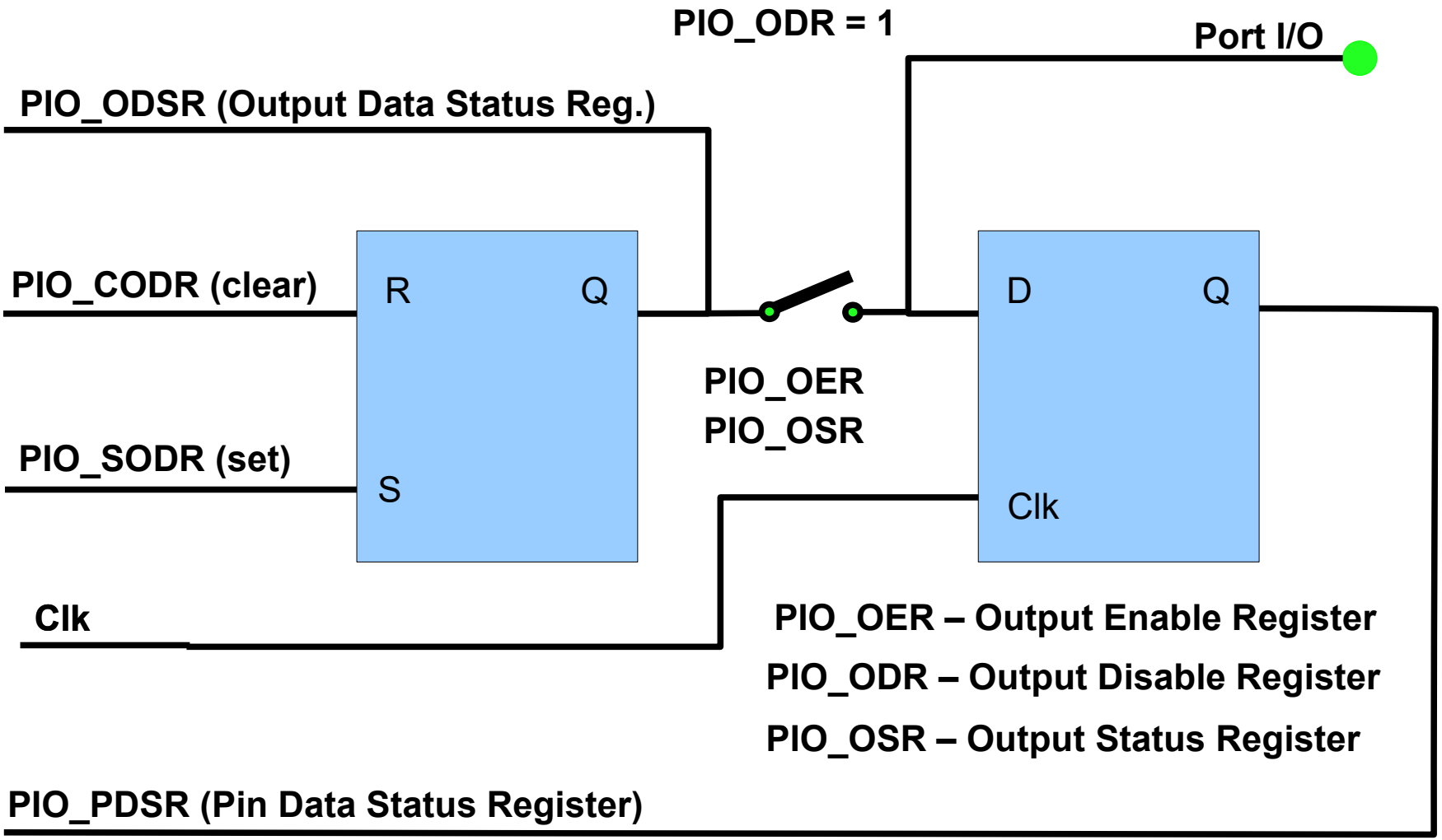
The Power Management Controller controls the PIO Controller clock in order to save power. Writing any of the registers of the user interface does not require the PIO Controller clock to be enabled. This means that the configuration of the I/O lines does not require the PIO Controller clock to be enabled.

However, when the clock is disabled, not all of the features of the PIO Controller are available. Note that the Input Change Interrupt and the read of the pin level require the clock to be validated.

After a hardware reset, the PIO clock is disabled by default.

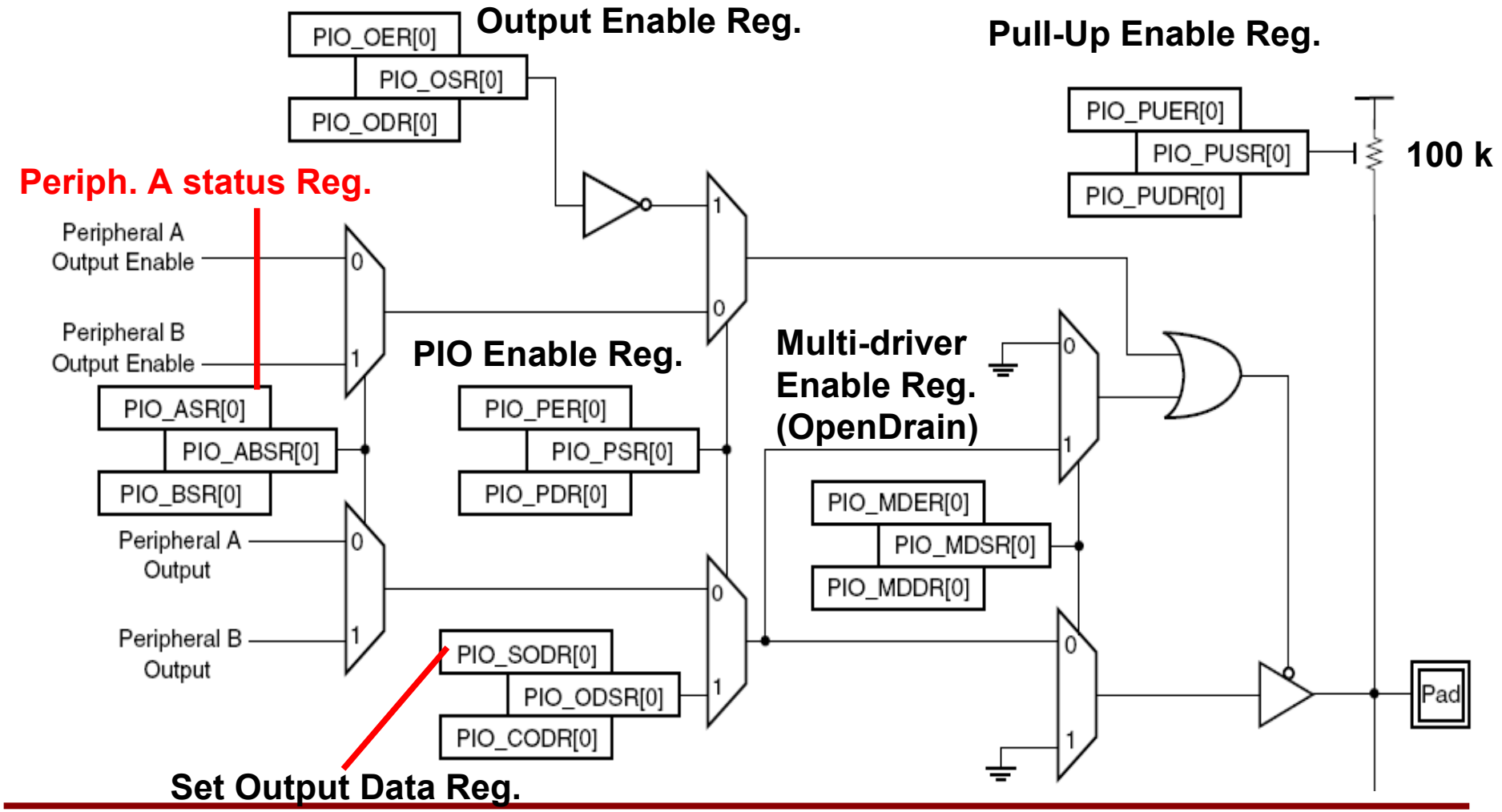
The user must configure the Power Management Controller before any access to the input line information.

Uproszczony schemat portu IO



Port IO – jak skonfigurować parametry portu ?

Figure 31-3. I/O Line Control Logic

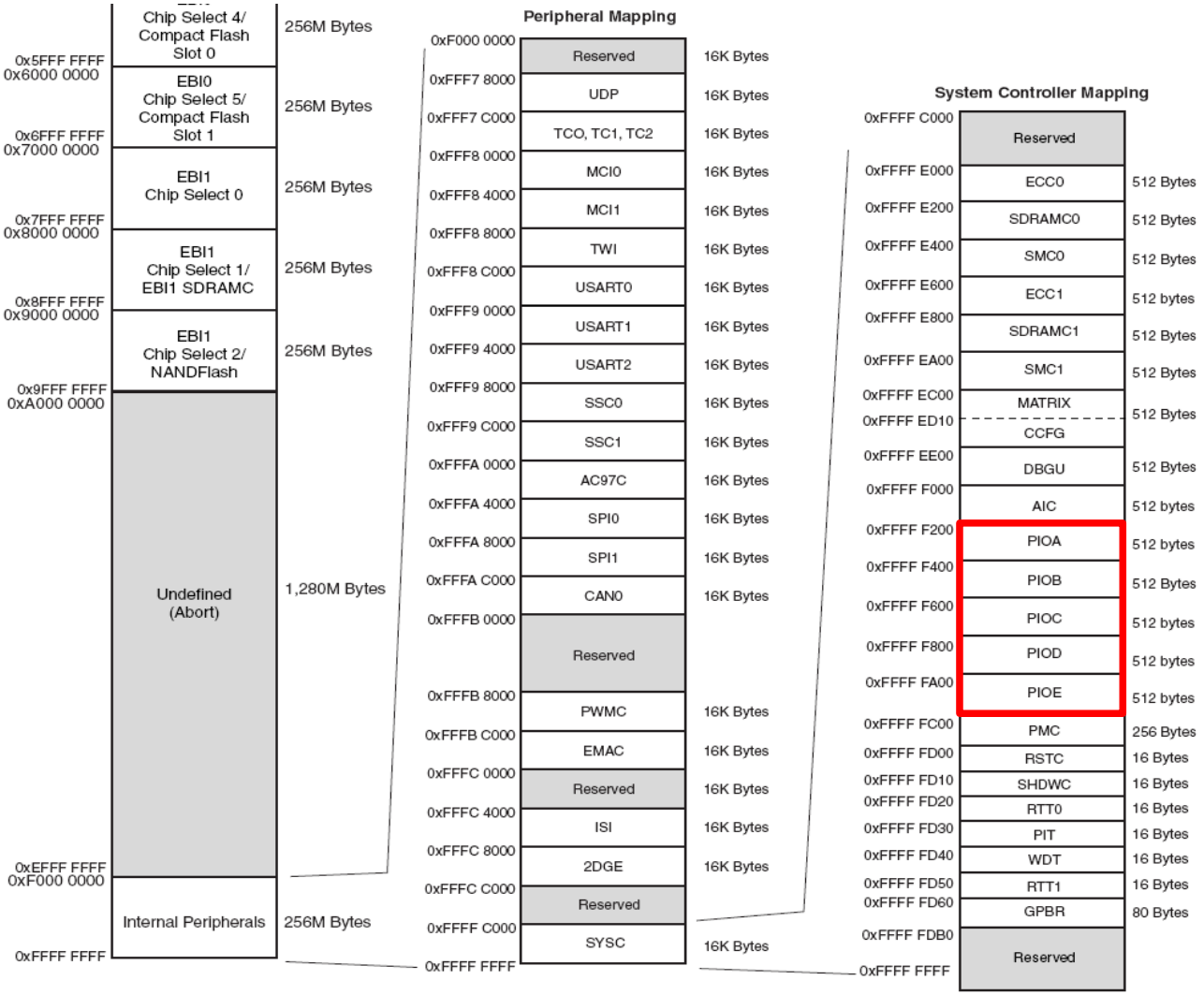


Control Registers for I/O ports

Table 31-2. Register Mapping

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			

Memory Map


Chip Select 4/ Compact Flash Slot 0
256M Bytes
0x5FFF FFFF
0x6000 0000
EBI0 Chip Select 5/ Compact Flash Slot 1
256M Bytes
0x6FFF FFFF
0x7000 0000
EBI1 Chip Select 0
256M Bytes
0x7FFF FFFF
0x8000 0000
EBI1 Chip Select 1/ EBI1 SDRAMC
256M Bytes
0x8FFF FFFF
0x9000 0000
EBI1 Chip Select 2/ NANDFlash
256M Bytes
0x9FFF FFFF
0xA000 0000
Undefined (Abort)
1,280M Bytes
0xEFFF FFFF
0xF000 0000
Internal Peripherals
256M Bytes
0xFFFF FFFF
Peripheral Mapping
0xF000 0000
Reserved
16K Bytes
0xFFF7 8000
UDP
16K Bytes
0xFFF7 C000
TCO, TC1, TC2
16K Bytes
0xFFF8 0000
MCI0
16K Bytes
0xFFF8 4000
MCI1
16K Bytes
0xFFF8 8000
TWI
16K Bytes
0xFFF8 C000
USART0
16K Bytes
0xFFF9 0000
USART1
16K Bytes
0xFFF9 4000
USART2
16K Bytes
0xFFF9 8000
SSC0
16K Bytes
0xFFF9 C000
SSC1
16K Bytes
0xFFFA 0000
AC97C
16K Bytes
0xFFFA 4000
SPI0
16K Bytes
0xFFFA 8000
SPI1
16K Bytes
0xFFFA C000
CAN0
16K Bytes
0xFFFB 0000
Reserved
0xFFFB 8000
PWMC
16K Bytes
0xFFFB C000
EMAC
16K Bytes
0xFFFC 0000
Reserved
16K Bytes
0xFFFC 4000
ISI
16K Bytes
0xFFFC 8000
2DGE
16K Bytes
0xFFFC C000
Reserved
0xFFFF C000
SYSC
16K Bytes
0xFFFF FFFF
System Controller Mapping
0xFFFF C000
Reserved
0xFFFF E000
ECC0
512 Bytes
0xFFFF E200
SDRAMC0
512 Bytes
0xFFFF E400
SMC0
512 Bytes
0xFFFF E600
ECC1
512 bytes
0xFFFF E800
SDRAMC1
512 Bytes
0xFFFF EA00
SMC1
512 Bytes
0xFFFF EC00
MATRIX
0xFFFF ED10
CCFG
512 Bytes
0xFFFF EE00
DBGU
512 Bytes
0xFFFF F000
AIC
512 bytes
0xFFFF F200
PIOA
512 bytes
0xFFFF F400
PIOB
512 Bytes
0xFFFF F600
PIOC
512 bytes
0xFFFF F800
PIOD
512 bytes
0xFFFF FA00
PIOE
512 bytes
0xFFFF FC00
PMC
256 Bytes
0xFFFF FD00
RSTC
16 Bytes
0xFFFF FD10
SHDWC
16 Bytes
0xFFFF FD20
RTT0
16 Bytes
0xFFFF FD30
PIT
16 Bytes
0xFFFF FD40
WDT
16 Bytes
0xFFFF FD50
RTT1
16 Bytes
0xFFFF FD60
GPBR
80 Bytes
0xFFFF FDB0
Reserved
0xFFFF FFFF

Rejestry odwzorowane w strukturze (1)

```
typedef volatile unsigned int *AT91_REG;      // Hardware register definition
        AT91_REG  PIO_PER = 0xFFFFF200;       // PIO Enable Register, 32-bit register
        AT91_REG  PIO_PDR = 0xFFFFF204;       // PIO Disable Register
        AT91_REG  PIO_PSR = 0xFFFFF208;       // PIO Status Register
        AT91_REG  Reserved0[1]= 0xFFFFF20C; // Filler
        AT91_REG  PIO_OER;                    // Output Enable Register
        AT91_REG  PIO_ODR;                    // Output Disable Registerr
        AT91_REG  PIO_OSR;                    // Output Status Register
        AT91_REG  Reserved1[1];                       //
        AT91_REG  PIO_IFER;                   // Input Filter Enable Register
        AT91_REG  PIO_IFDR;                   // Input Filter Disable Register
        AT91_REG  PIO_IFSR;                   // Input Filter Status Register
        AT91_REG  Reserved2[1];                       //
        AT91_REG  PIO_SODR;                   // Set Output Data Register
        AT91_REG  PIO_CODR;                   // Clear Output Data Register
        AT91_REG  PIO_ODSR;                   // Output Data Status Register
```

Rejestry odwzorowane w strukturze (2)

```
// Struktura opisująca rejestry portów IO procesora ARM
typedef volatile unsigned int AT91_REG;          // Hardware register definition
typedef struct _AT91S_PIO {
        AT91_REG  PIO_PER;          // PIO Enable Register, 32-bit register
        AT91_REG  PIO_PDR;          // PIO Disable Register
        AT91_REG  PIO_PSR;          // PIO Status Register
        AT91_REG  Reserved0[1];             // 32-bit filler
        AT91_REG  PIO_OER;          // Output Enable Register
        AT91_REG  PIO_ODR;          // Output Disable Register
        AT91_REG  PIO_OSR;          // Output Status Register
        AT91_REG  Reserved1[1];             // 32-bit filler
        AT91_REG  PIO_IFER;         // Input Filter Enable Register
        AT91_REG  PIO_IFDR;         // Input Filter Disable Register
        AT91_REG  PIO_IFSR;         // Input Filter Status Register
        AT91_REG  Reserved2[1];             // 32-bit filler
        AT91_REG  PIO_SODR;         // Set Output Data Register
        AT91_REG  PIO_CODR;         // Clear Output Data Register
        AT91_REG  PIO_ODSR;         // Output Data Status Register
} AT91S_PIO, *AT91PS_PIO;
```

Rejestry odwzorowane w strukturze (1)

Deklaracja nowego typu danych tworzy szablon układu rejestrów procesora w pamięci. Rejestrom zostają przypisane nazwy symboliczne. Utworzono nowy typ danych **AT91S_PIO** oraz wskaźnik ***AT91PS_PIO** na ten typ.

Brak informacji o dostępie do rejestrów (R/W) oraz wartości rejestrów po resecie.

W celu uzupełnienia brakujących informacji można umieścić dodatkowe komentarze.

```
typedef struct _AT91S_PIO {          /*     Register name              R/W     Reset val.     Offset
        AT91_REG  PIO_PER;           // PIO Enable Register             W       -              0x00
        AT91_REG  PIO_PDR;           // PIO Disable Register            W       -              0x04
        AT91_REG  PIO_PSR;           // PIO Status Register             R       0x0            0x08
        AT91_REG  Reserved0[1];              //
        AT91_REG  PIO_OER;           // Output Enable Register          W       -              0x10
        AT91_REG  PIO_ODR;           // Output Disable Register         W       -              0x14
        AT91_REG  PIO_OSR;           // Output Status Register          W       -              0x18
        }
        /* blok rejestrów portów I/O PIOA...PIOE */
#define AT91C_BASE_PIOA      (AT91PS_PIO)    0xFFFFF200     // (PIOA) Base Address
        /* maska zerowego bitu portu PA */
#define AT91C_PIO_PA0        (1 <<  0)        // Pin Controlled by PA0
```

Jak ustawić bity 0 i 19 rejestru PIO_PER ?

Wskaźniki do rejestrów portów I/O PIOA...PIOE

```
/* wskaźniki do rejestrów portów I/O PIOA...PIOE */
#define AT91C_BASE_AIC      (AT91_CAST(AT91PS_AIC)      0xFFFFF000) // (AIC) Base Address
#define AT91C_BASE_PIOA     (AT91_CAST(AT91PS_PIO)      0xFFFFF200) // (PIOA) Base Address
#define AT91C_BASE_PIOB     (AT91_CAST(AT91PS_PIO)      0xFFFFF400) // (PIOB) Base Address
#define AT91C_BASE_PIOC     (AT91_CAST(AT91PS_PIO)      0xFFFFF600) // (PIOC) Base Address
#define AT91C_BASE_PIOD     (AT91_CAST(AT91PS_PIO)      0xFFFFF800) // (PIOD) Base Address
#define AT91C_BASE_PIOE     (AT91_CAST(AT91PS_PIO)      0xFFFFFA00) // (PIOE) Base Address
#define AT91C_BASE_CKGR     (AT91_CAST(AT91PS_CKGR)     0xFFFFFC20) // (CKGR) Base Address
#define AT91C_BASE_PMC      (AT91_CAST(AT91PS_PMC)      0xFFFFFC00) // (PMC) Base Address
```

Odwołanie do rejestrów

Zapis do rejestru

```
AT91PS_PIO->PIO_OER = 0x5;
```

Odczyt danej z rejestru

```
volatile unsigned int ReadData;
ReadData = AT91PS_PIO->PIO_OSR;
```

Operacje bitowe

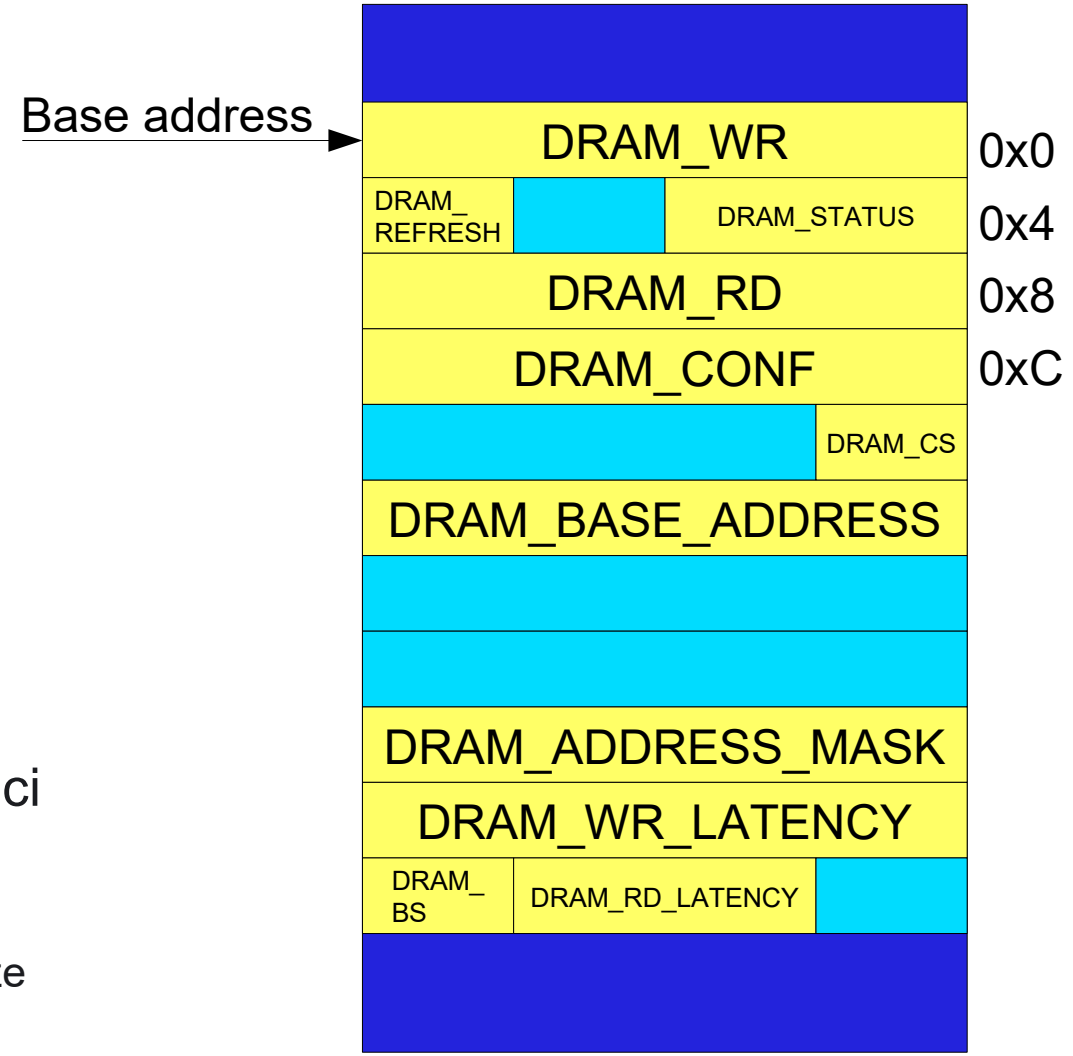
```
AT91C_BASE_PIOA->PIO_PER  = (AT91C_PIO_PA0 | AT91C_PIO_PA19);
AT91C_BASE_PIOA->PIO_PDR = (AT91C_PIO_PA0 | AT91C_PIO_PA19);
AT91C_BASE_PIOA->PIO_ODSR ^= (AT91C_PIO_PA0 | AT91C_PIO_PA19);
```

Rejestry odwzorowane w strukturze - ćwiczenie

- Rejestry pamięci DRAM są mapowane na przestrzeń pamięci,
- Adres bazowy: 0xFFFE.2000,
- Typ rejestrów: 8, 16, 32 bit

Zadanie do wykonania:

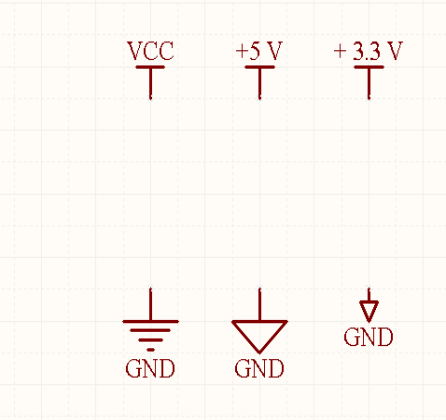
- Utwórz nowy typ struktury dla rejestrów DRAM,
- Zadeklaruj wskaźnik,
- Odczytaj, zapisz dane z pamięci
  - Ustaw i wyczyść bity rejestru DRAM_CONF (bit 5, bit 29),
  - Sprawdź flagę zajętości w rejestrze stanu DRAM_STATUS (bit 9)



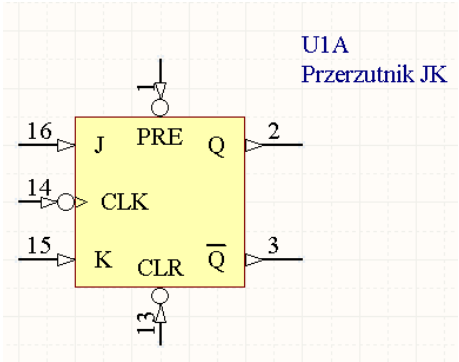
Schematy elektryczne

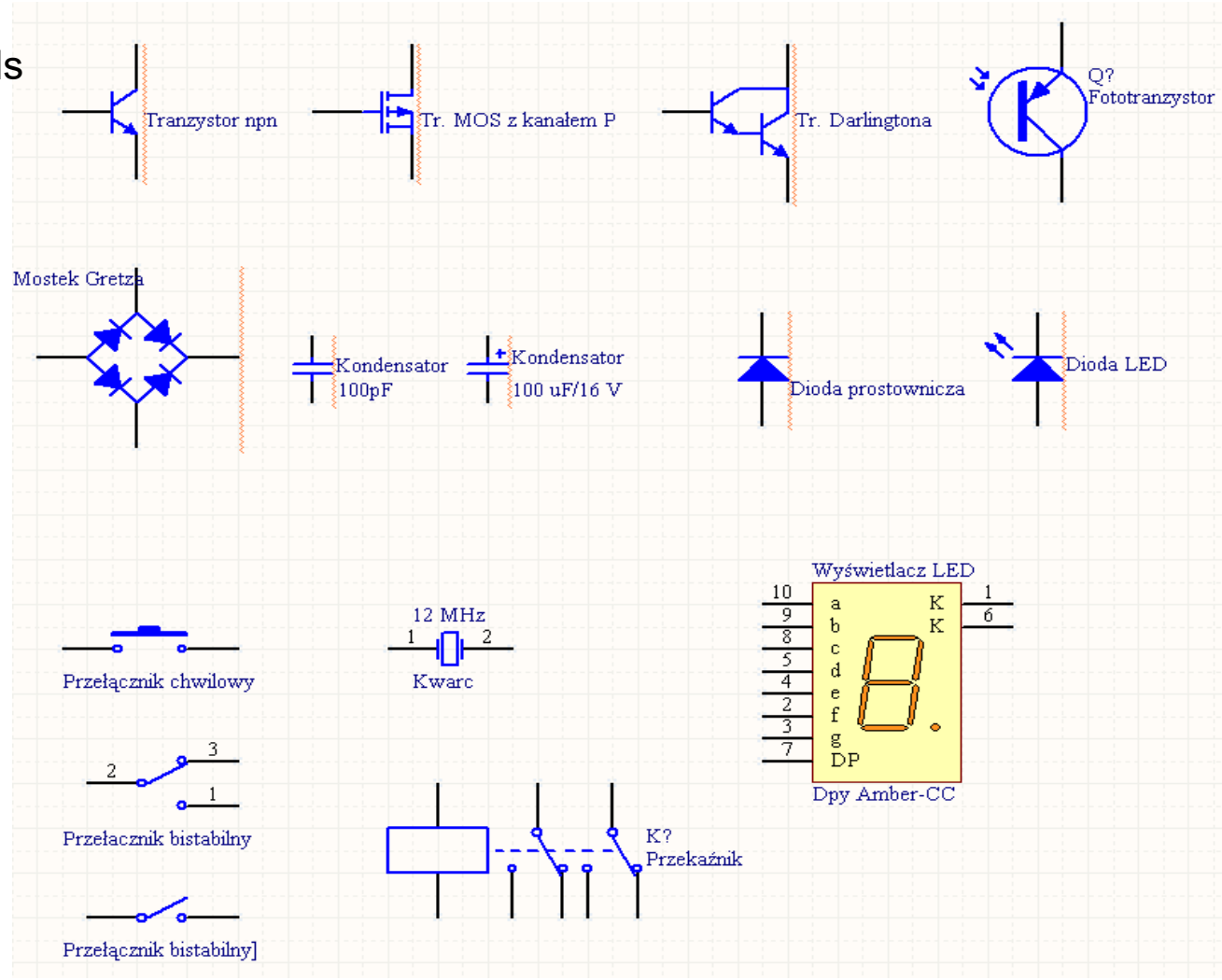
Schematic Diagrams (1)

Power Supply Bus Symbols



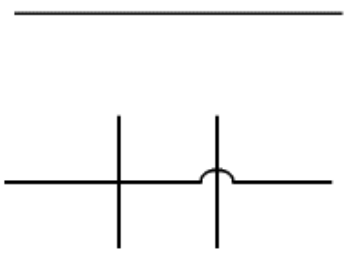
Ground Symbols

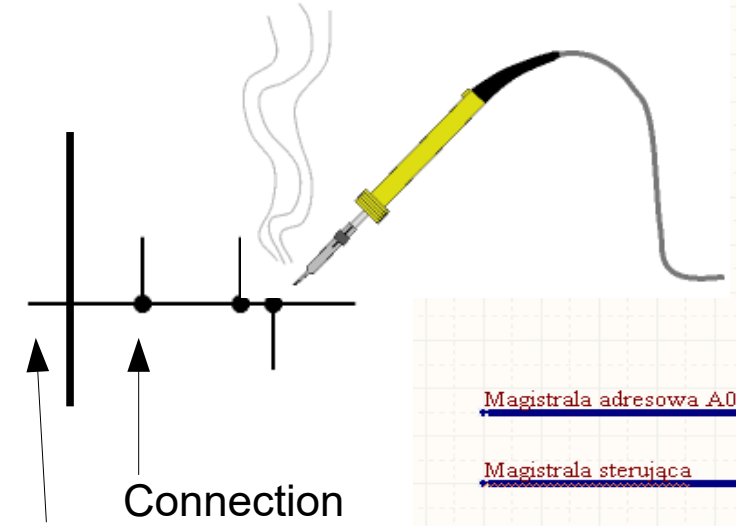




Schematic Diagrams (2)

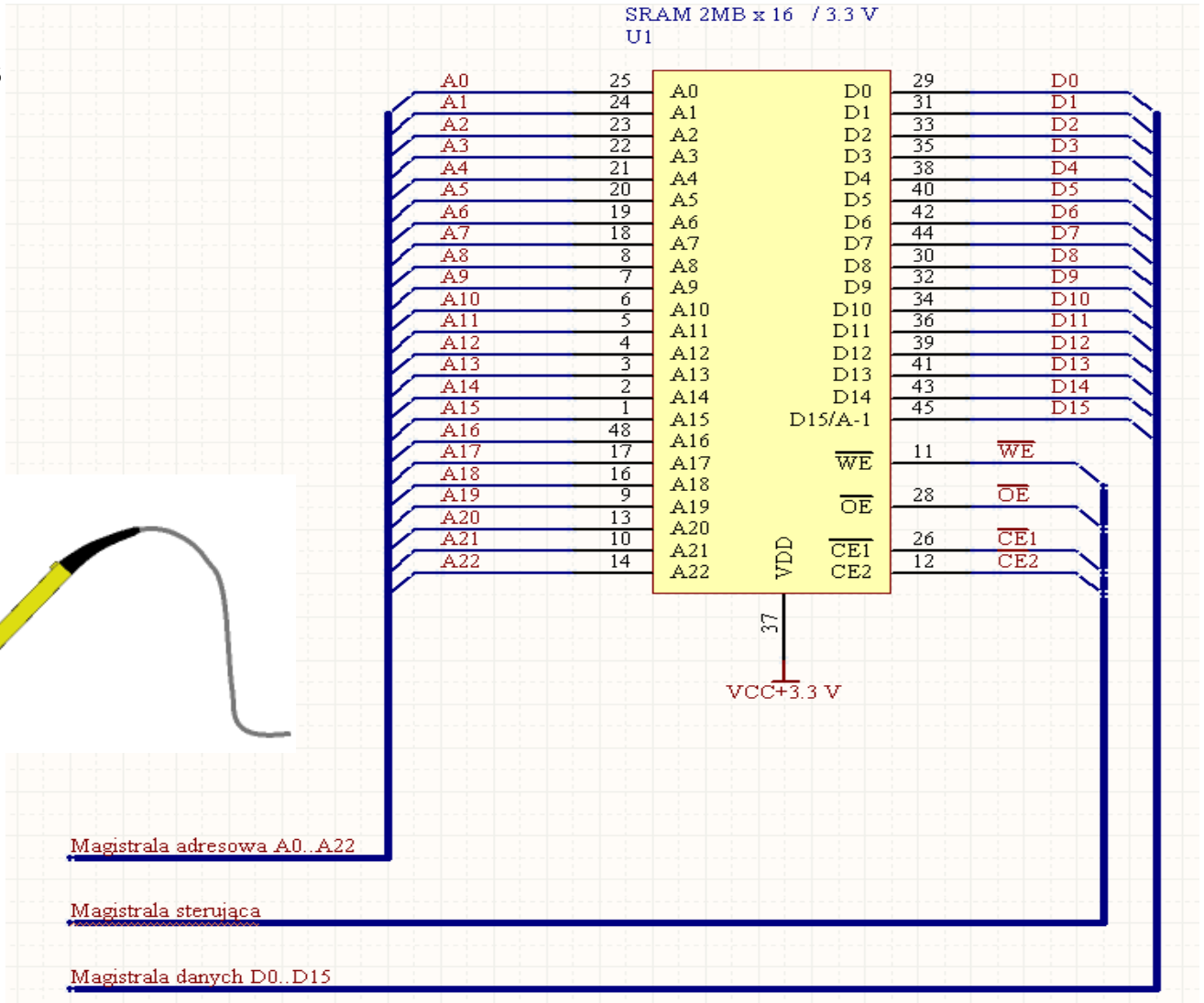
Electrical connections



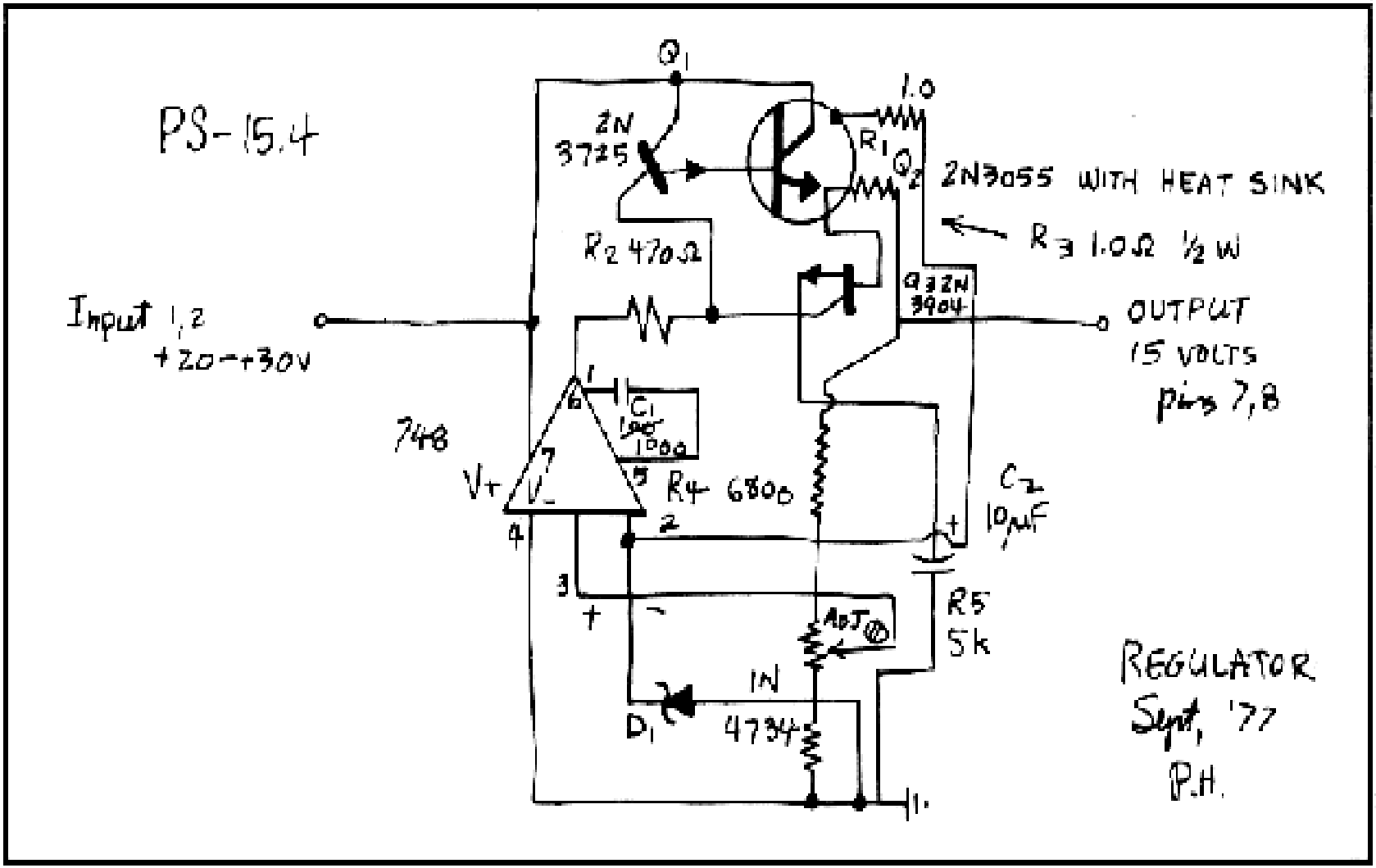


Connection

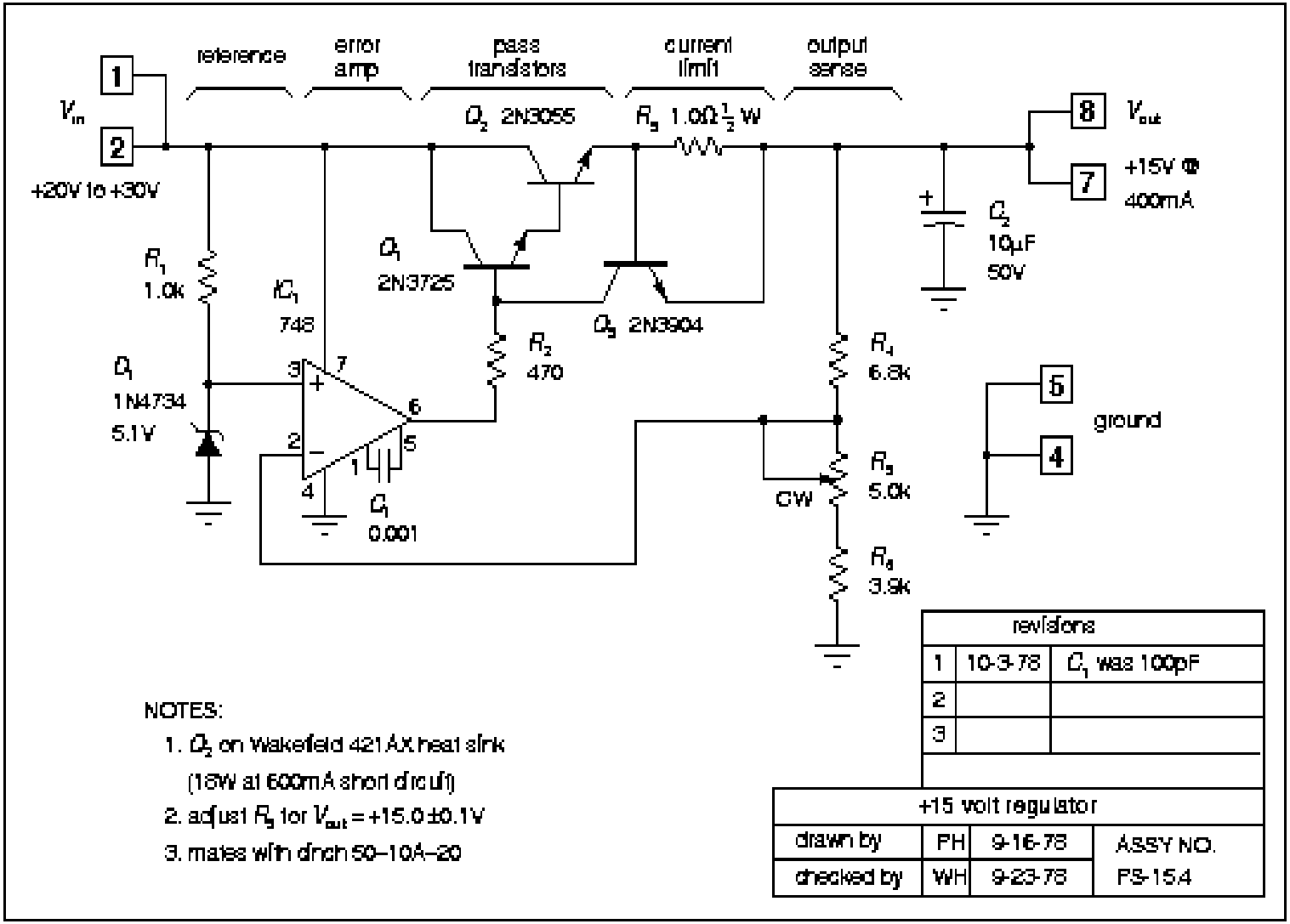
No connection



Schematic Diagram – How to Draw ?



Schematic Diagrams – Better Way



NOTES:

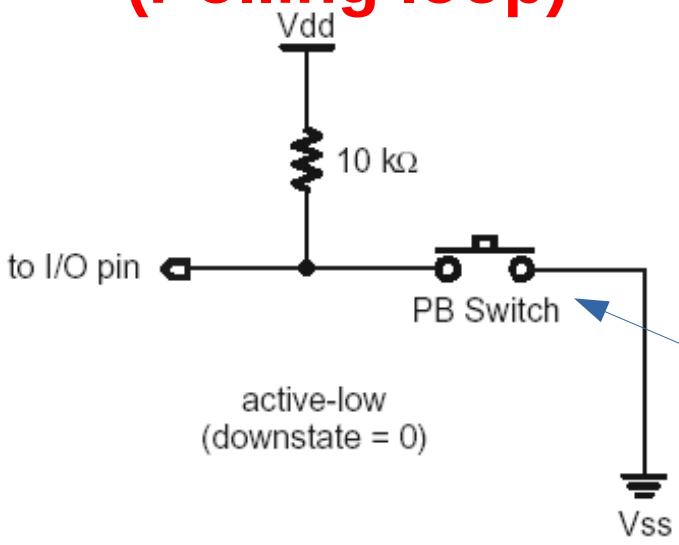
1. Q_2 on Wakefield 421AK heat sink (16W at 600mA short circuit)
2. adjust R_5 for V_{out} = +15.0 ±0.1V
3. mates with dinch 50–10A–20

revisions		
1	10-3-78	C_1 was 100pF
2		
3		

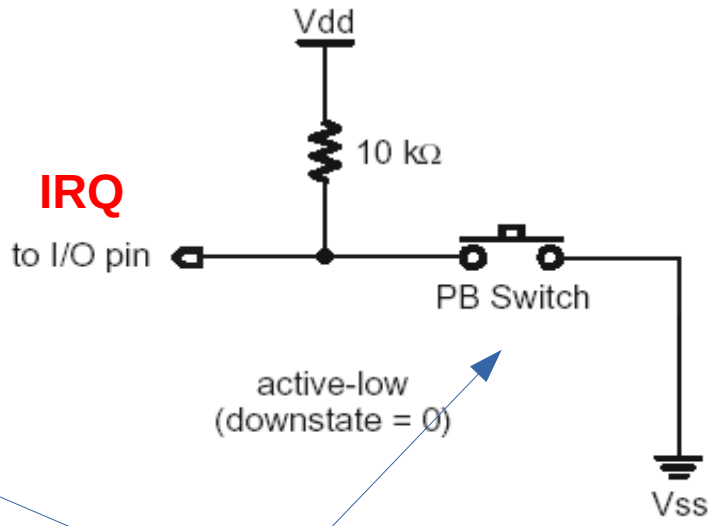
+15 volt regulator			
drawn by	PH	9-16-78	ASSY NO.
checked by	WH	9-23-78	PS-154

Podłaczenie Przycisków

Odpytywanie (Polling loop)

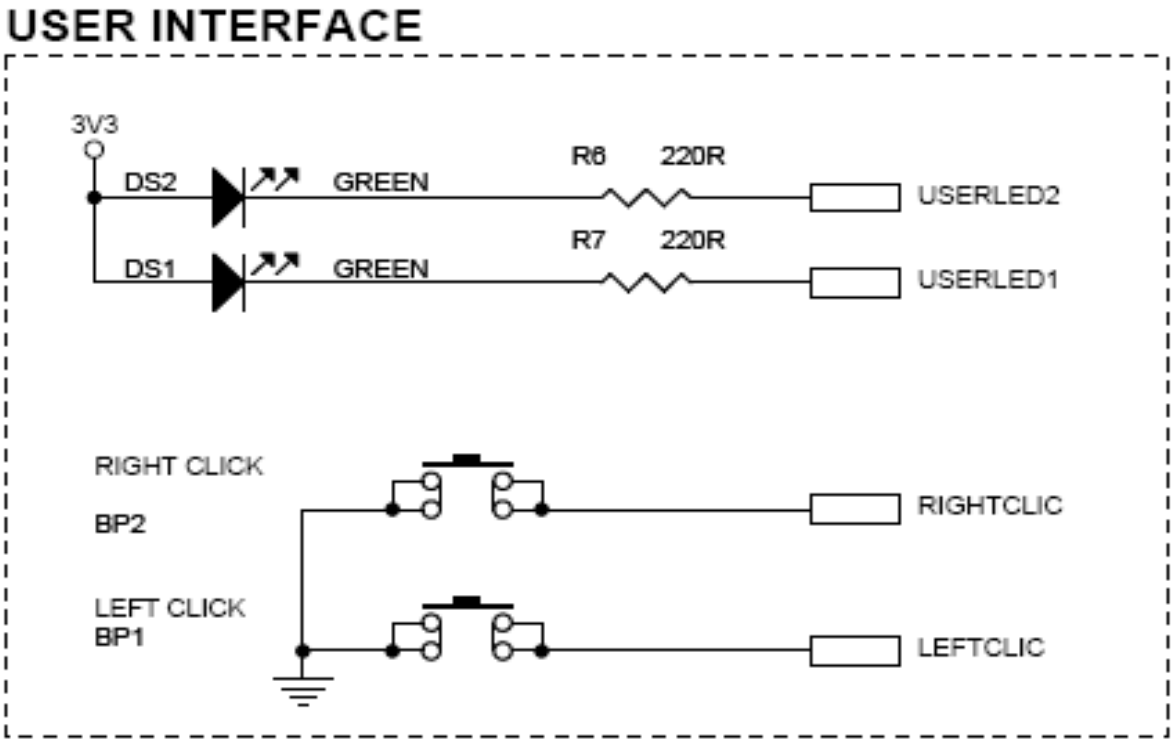


Przerwanie



Asynchronous signal

Płyta uruchomieniowa MSC – diody LED, klawiatura




```
#define AT91B_LED1      AT91C_PIO_PB8   /* DS1 */
#define AT91B_LED2      AT91C_PIO_PC29  /* DS2 */
#define AT91B_BP1       AT91C_PIO_PC5  // Left click
#define AT91B_BP2       AT91C_PIO_PC4  // Right clic
```