KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

”Procesory ARM w systemach wbudowanych”

”Rodzina procesorów ARM”

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”

Prezentacja dystrybuowana jest bezpłatnie

Dariusz Makowski

Katedra Mikroelektroniki i Technik Informatycznych

tel. 631 2720

dmakow@dmcs.pl

http://neo.dmcs.p.lodz.pl/arm/

ul. Wólczańska 221/223, budynek B18, pokój 40

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Zakres przedmiotu

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

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Od Acorn Computers Ltd. ARM do ARM Ltd.

Acorn

- Firma utworzona w 1990 roku,
- Powstała z firmy produkującej komputery Acorn Computers,
- ***Firma ARM nie produkuje procesorów***,
- Projektuje i sprzedaje licencje projektów tzw. rdzeni (**I**ntellectual **P**roperty **C**ores) procesorów ARM oraz urządzeń peryferyjnych
- Produkuje narzędzia, płyty prototypowe, narzędzia do debugowania, standardy

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wybrane firmy produkujące procesory zgodne z architekturą ARM

Agi-lent, AKM, Alcatel, Altera, Atmel, Broadcom, Chip Express, Cirrus Logic, Digital Semiconductor, eSilicon, Fujitsu, GEC Plessey, Global UniChip, HP, Hyundai, IBM, Intel, ITRI, LG Semicon, LSI Logic, Lu-cent, Matsushita, Micrel, Micronas, Mitsubishi, Freescale, NEC, OKI, Philips, Qu-alcomm, Rockwell, Rohm, Samsung, Samsung, Sanyo, Seagate, Seiko Epson, Sharp, Sony, STMicroelectronics, Symbios Logic, Texas Instru-ments, Xilinx, Yamaha, Zeevo, ZTEIC, ...

Historia mikroprocesorów ARM

1983 – Sophie Wilson and Steve Furber projektują pierwszy procesor RISC w firmie Acorn Computers Limited, Cambridge, ARM = **A**corn (**A**dvanced) **R**ISC **M**achine

1985 – prototypowe procesory ARM 1 (wersja architektury v1)

1986 – procesory ARM 2 opuszczają fabrykę (32-bit, 26-bit adres, 16 rejestrów 16 bitowych, 30.000 tranzystorów, wersja architektury v2/v2a, 8 MHz)

1980 – Apple Computer i VLSI Technology rozpoczynają współpracę nad kolejną wersja rdzenia procesora ARM

1990 – wyłania się nowa firma Advanced RISC Machines Ltd. odpowiedzialna za dalszy rozwój procesorów ARM

1991 – pojawia się pierwszy procesor będący wynikiem współpracy z firmą Apple – procesor z jądrem ARM 6 (stosowany w jako procesor ARM 610 w Apple Newton PDA, wersja architektury v3, 33 MHz)

1995 – firma ARM wprowadza na rynek bardzo udany model procesora z rdzeniem **ARM7TDMI** (**architektura ARMv4T**) oraz StrongARM (233 MHz)

2001 – firma ARM wprowadza kolejny model procesora z rdzeniem **ARM9TDMI** (**architektura ARMv5TEJ**, 220 MHz)

2004 – procesor z rdzeniem Cortex M3 (ARMv7-M, 100 MHz)

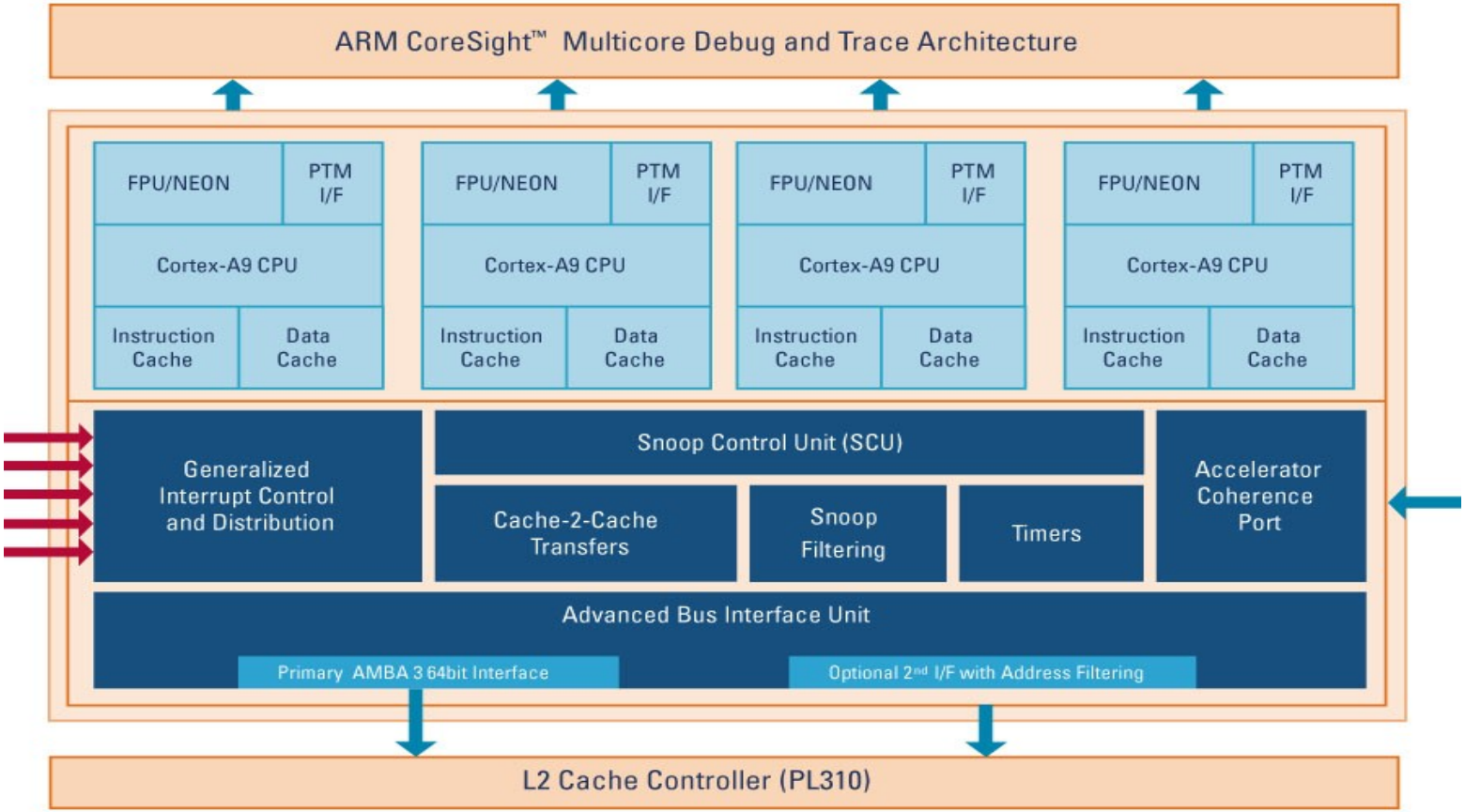
2008 – kolejny model procesora ARM Cordex A8 (architektura ARMv7, 1 GHz)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

ARM Cortex A9 w konfiguracji MPCore



Technologia MPCore pozwala na budowę SoC – cztery rdzenie procesorów A9

Procesory z rdzeniem ARM

- Procesory ARM są szeroko stosowane w systemach wbudowanych i systemach o niskim poborze mocy, ze względu na energooszczędną architekturę
- Procesor ARM jest jednym z najczęściej stosowanych procesorów na świecie. Jest używany w dyskach twardych, telefonach komórkowych, routerach, kalkulatorach a nawet w zabawkach dziecięcych
- Obecnie zajmuje ponad 75% rynku 32-bitowych CPU dla systemów wbudowanych
- Najbardziej udanym projektem ARM był procesor ARM7TDMI szeroko stosowany w telefonach komórkowych
- Moc obliczeniowa procesorów ARM umożliwia instalacje na tym procesorze systemu operacyjnego, z zaimplementowanymi mechanizmami wielowątkowości, z możliwością wykorzystania zawartego w systemie stosu TCP/IP, systemu plików (np. FAT32).
- Systemy operacje: dystrybucje embedded Linuk (Embedded Debian, Embedded Ubuntu), Windows CE, Symbian, NUTOS (Ethernut), ...

Zastosowanie procesorów ARM

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Porównanie wybranych procesorów ARM

Family	Architecture Version	Core	Feature	Cache (I/D)/MMU	Typical MIPS @ MHz
ARM6	ARMv3	ARM610	Cache, no coprocessor	4K unified	17 MIPS @ 20 MHz
ARM7	ARMv3	ARM7500FE	Integrated SoC. "FE" Added FPA and EDO memory controller.	4 KB unified	55 MIPS @ 56 MHz
ARM7TDMI	ARMv5TEJ	ARM7EJ-S	Jazelle DBX, Enhanced DSP instructions, 5-stage pipeline	8 KB	120 MIPS @ 133 MHz
StrongARM	ARMv4	SA-110	5-stage pipeline, MMU	16 KB/16 KB, MMU	235 MIPS @ 206 MHz
ARM8	ARMv4	ARM810[7]	5-stage pipeline, static branch prediction, double-bandwidth memory	8 KB unified, MMU	1.0 DMIPS/MHz
ARM9TDMI	ARMv4T	ARM920T	5-stage pipeline	16 KB/16 KB, MMU	245 MIPS @ 250 MHz
ARM9E	ARMv5TEJ	ARM926EJ-S	Jazelle DBX, Enhanced DSP instructions	variable, TCMs, MMU	220 MIPS @ 200 MHz
ARM10E	ARMv5TE	ARM1020E	VFP, 6-stage pipeline, Enhanced DSP instructions	32 KB/32 KB, MMU	300 MIPS @ 325 MHz
XScale	ARMv5TE	PXA27x	MMX and SSE instruction set, four MACs,	32 Kb/32 Kb, MMU	800 MIPS @ 624 MHz
ARM11	ARMv6	ARM1136J(F)-S	SIMD, Jazelle DBX, VFP, 8-stage pipeline	variable, MMU	740 @ 532-665 MHz
Cortex	ARMv7-A	Cortex-A8	Application profile, VFP, NEON, Jazelle RCT, Thumb-2, 13-stage superscalar pipeline	variable (L1+L2), MMU+TrustZone	>1000 MIPS@ 600 M-1 GHz

Systemy liczbowe

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Systemy liczbowe

W systemach komputerowych wykorzystujemy następujące systemy liczbowe:

- dziesiętny DEC (cyfry od 0-9),
- dwójkowy BIN (cyfry 0,1),
- ósemkowy OCT (cyfry 0-7),
- szesnastkowy HEX (cyfry 0-F).

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

System dwójkowy

Dwójkowy system liczbowy (inaczej binarny) to pozycyjny system liczbowy, w którym podstawą jest liczba 2.

$\mathbf{1010}_{\mathbf{bin}}$ =

$1x\ 2^3 + 0x\ 2^2 + 1x\ 2^1 + 0x\ 2^0 = 8 + 2 = \mathbf{1}$

$01010100101111011010101010101011010110 10_b$ = ???

$0101.0100.1011.1101.1010.1010.1010.1011.0101.1010_b$

System szesnastkowy

Szesnastkowy system liczbowy (czasem nazywany heksadecymalnym, skrót hex) – pozycyjny system liczbowy, w którym podstawą jest liczba 16.

$AB98_{hex}$ =

$10x\ 16^3 + 11x\ 16^2 + 9x\ 16^1 + 8x\ 16^0 =$

$40960 + 2816 + 144 + 8 = \mathbf{43928}_{dec}$

0_{hex}	0_{dec}	0_{oct}	0	0	0	0
1_{hex}	1_{dec}	1_{oct}	0	0	0	1
2_{hex}	2_{dec}	2_{oct}	0	0	1	0
3_{hex}	3_{dec}	3_{oct}	0	0	1	1
4_{hex}	4_{dec}	4_{oct}	0	1	0	0
5_{hex}	5_{dec}	5_{oct}	0	1	0	1
6_{hex}	6_{dec}	6_{oct}	0	1	1	0
7_{hex}	7_{dec}	7_{oct}	0	1	1	1
8_{hex}	8_{dec}	10_{oct}	1	0	0	0
9_{hex}	9_{dec}	11_{oct}	1	0	0	1
A_{hex}	10_{dec}	12_{oct}	1	0	1	0
B_{hex}	11_{dec}	13_{oct}	1	0	1	1
C_{hex}	12_{dec}	14_{oct}	1	1	0	0
D_{hex}	13_{dec}	15_{oct}	1	1	0	1
E_{hex}	14_{dec}	16_{oct}	1	1	1	0
F_{hex}	15_{dec}	17_{oct}	1	1	1	1

$0xABCCDEF09 \rightarrow ABCDEF09_{hex}$

$\rightarrow ???_{dec} \rightarrow ???_{bin}$

0xAB.CD.EF.09 0xABCD.EF09 $ABCD|EF09_{hex}$

Zamiana systemów liczbowych

$$1011.1101.1010.1010.1010.1011.0101.1010_{bin}$$

$$B\ .\ D\ .\ A\ .\ A\ .\ A\ .\ B\ .\ 5\ .\ A_{hex}$$

0xBD.AA.AB.5A

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

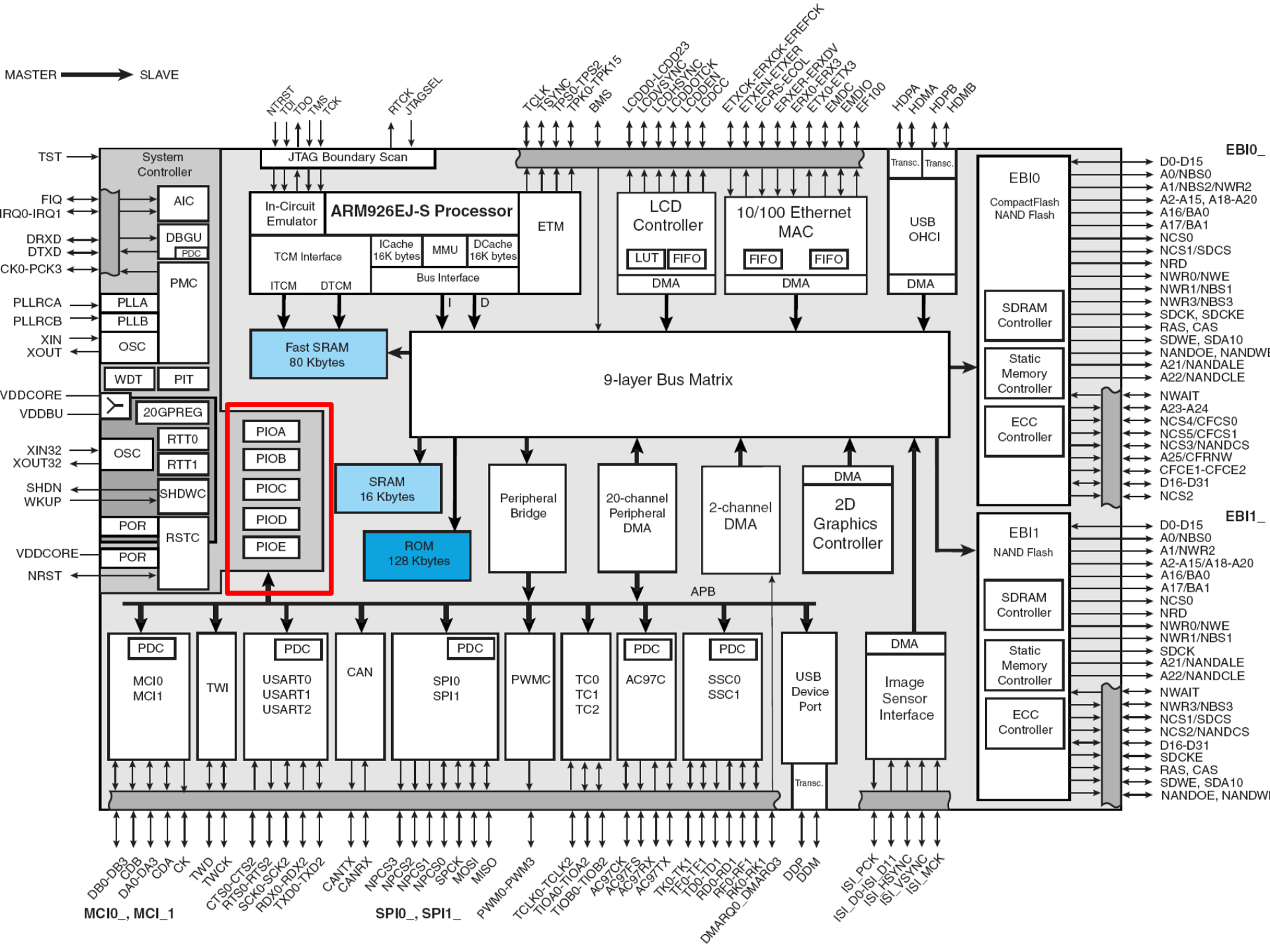
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Porty wejścia-wyjścia procesora ARM AT91SAM9263

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY


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
FIQ
IRQ0-IRQ1
DRXD
DTXD
CK0-PCK3
PLLRCA
PLLRCB
XIN
XOUT
VDDCORE
VDDBU
XIN32
XOUT32
SHDN
WKUP
VDDCORE
NRST
System Controller
AIC
DBGU
PDC
PMC
PLLA
PLLB
OSC
WDT
PIT
20GPREG
OSC
RTT0
RTT1
SHDWC
POR
RSTC
POR
PIOA
PIOB
PIOC
PIOD
PIOE
JTAG Boundary Scan
In-Circuit Emulator
ARM926EJ-S Processor
ETM
TCM Interface
ICache 16K bytes
MMU
DCache 16K bytes
Bus Interface
ITCM
DTCM
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
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_

Mikrokontroler AT91SAM9263 - dokumentacja

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

Mikrokontroler AT91SAM9263 – porty I/O

AT91SAM9263 Preliminary

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:

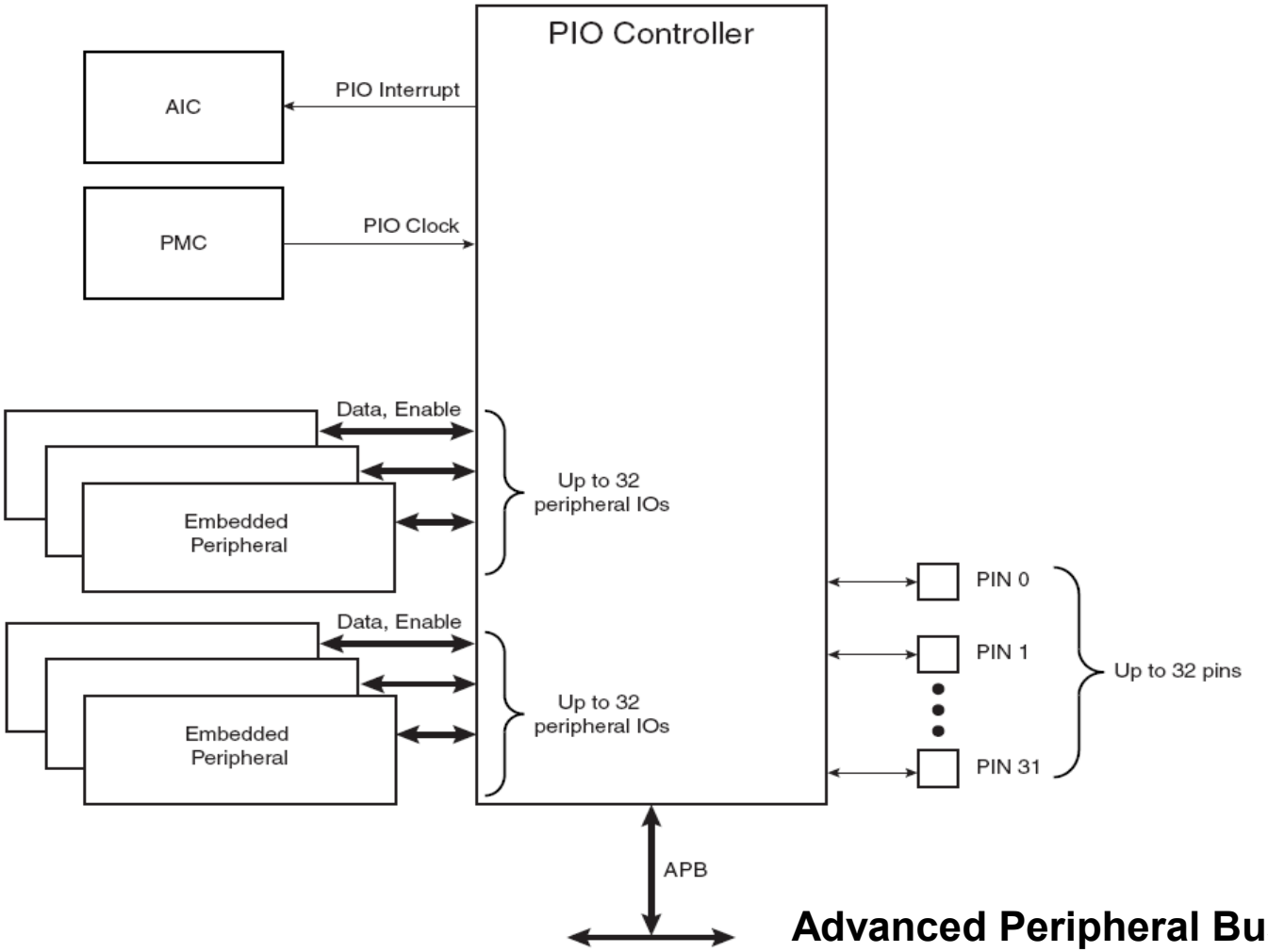
- An input change interrupt enabling level change detection on any I/O line.
- A glitch filter providing rejection of pulses lower than one-half of clock cycle.
- Multi-drive capability similar to an open drain I/O line.
- Control of the the pull-up of the I/O line.
- Input visibility and output control.

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

Schemat blokowy 32-bitowego portu I/O

Figure 31-1. Block Diagram



Zegar, a pobór energii

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.

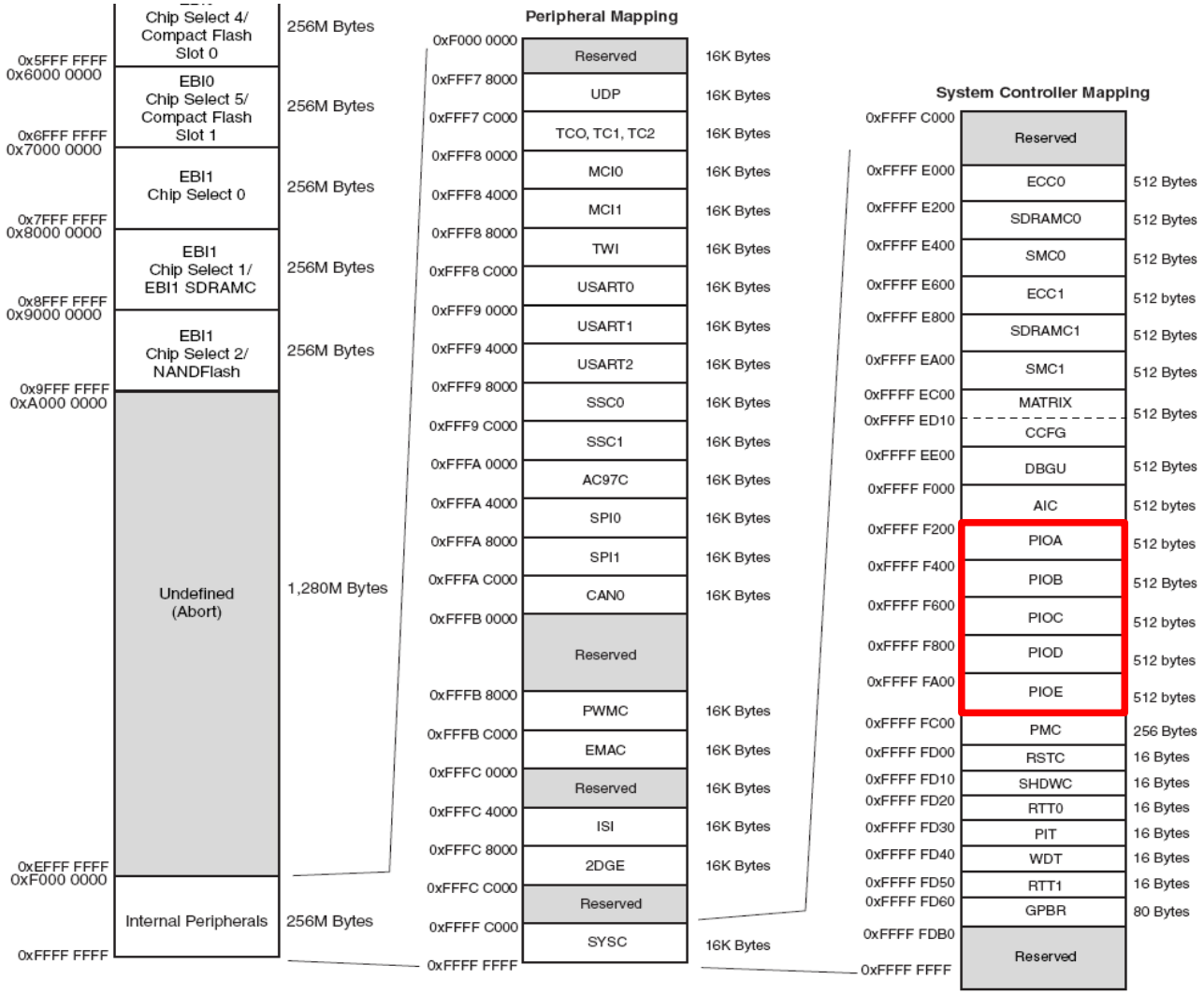
The user must configure the Power Management Controller before any access to the input line information.

Rejestry sterujące portem I/O

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			

Mapa pamięci rejestrów



Rejestry procesora

Rejestry procesora - komórki pamięci o niewielkich rozmiarach (najczęściej 4/8/16/32/64/128/256/512 bitów) umieszczone wewnątrz procesora i służące do przechowywania tymczasowych wyników obliczeń, adresów lokacji w pamięci operacyjnej itd. Procesory wykonują działania wyłącznie korzystając z wewnętrznych rejestrów, kopiując do nich dane z pamięci i po zakończeniu obliczeń odsyłając wynik do pamięci.

Rejestry procesora stanowią najwyższy szczebel w hierarchii pamięci, będąc najszybszym z rodzajów pamięci komputera, zarazem najdroższą w produkcji, a co za tym idzie - o najmniejszej pojemności. Realizowane zazwyczaj za pomocą przerzutników dwustanowych, z reguły jako tablica rejestrów (blok rejestrów, z ang. register file).

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Opis rejestów w dokumentacji

31.6.12 PIO Controller Output Data Status Register

Name: PIO_ODSR

Addresses: 0xFFFFF238 (PIOA), 0xFFFFF438 (PIOB), 0xFFFFF638 (PIOC), 0xFFFFF838 (PIOD), 0xFFFFFA38 (PIOE)

Access Type: Read-only or Read-write

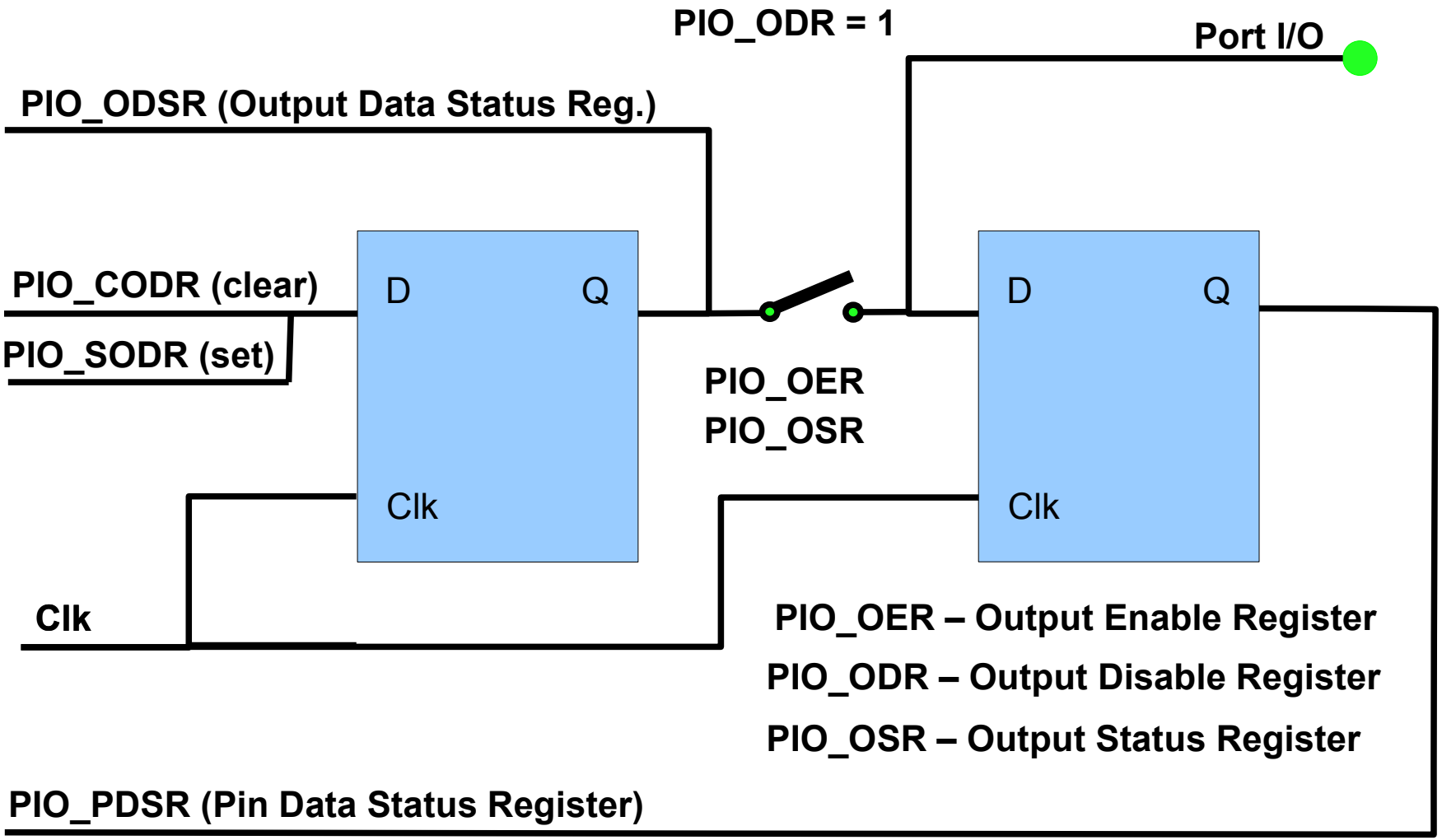
31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0

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

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

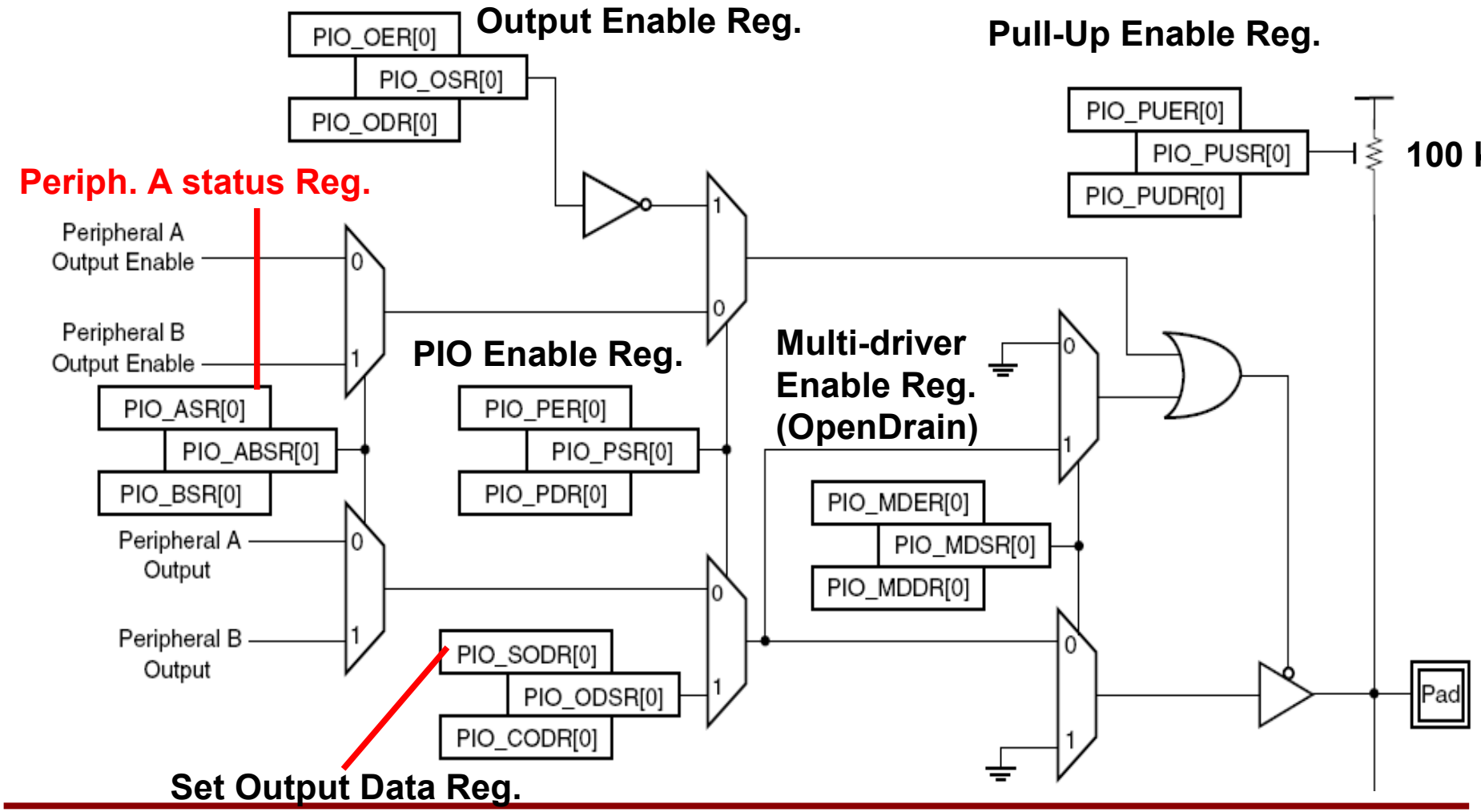
1 = The data to be driven on the I/O line is 1.

Schemat blokowy portu I/O

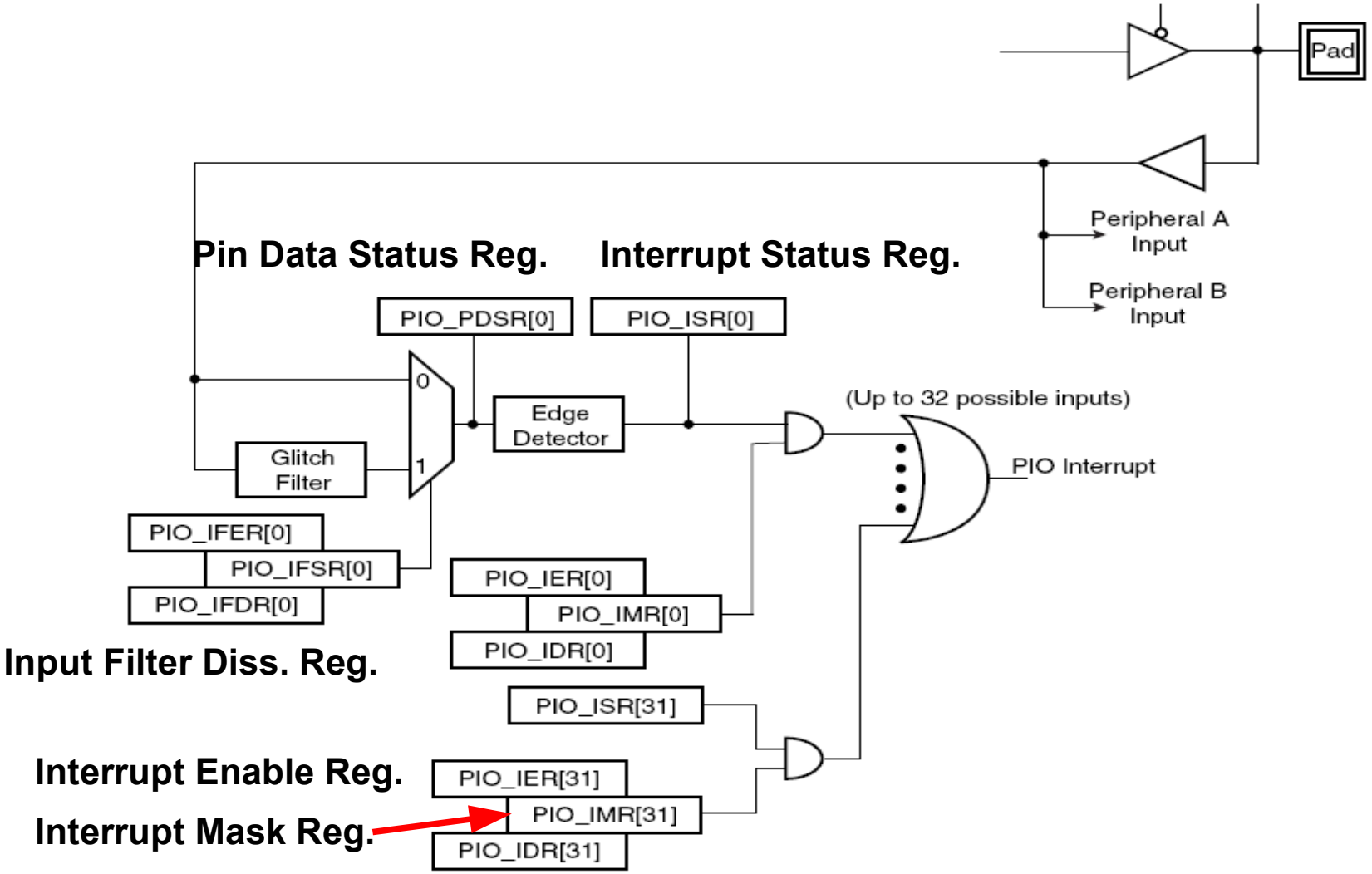


Schemat blokowy portu I/O – sterowanie wyjściem

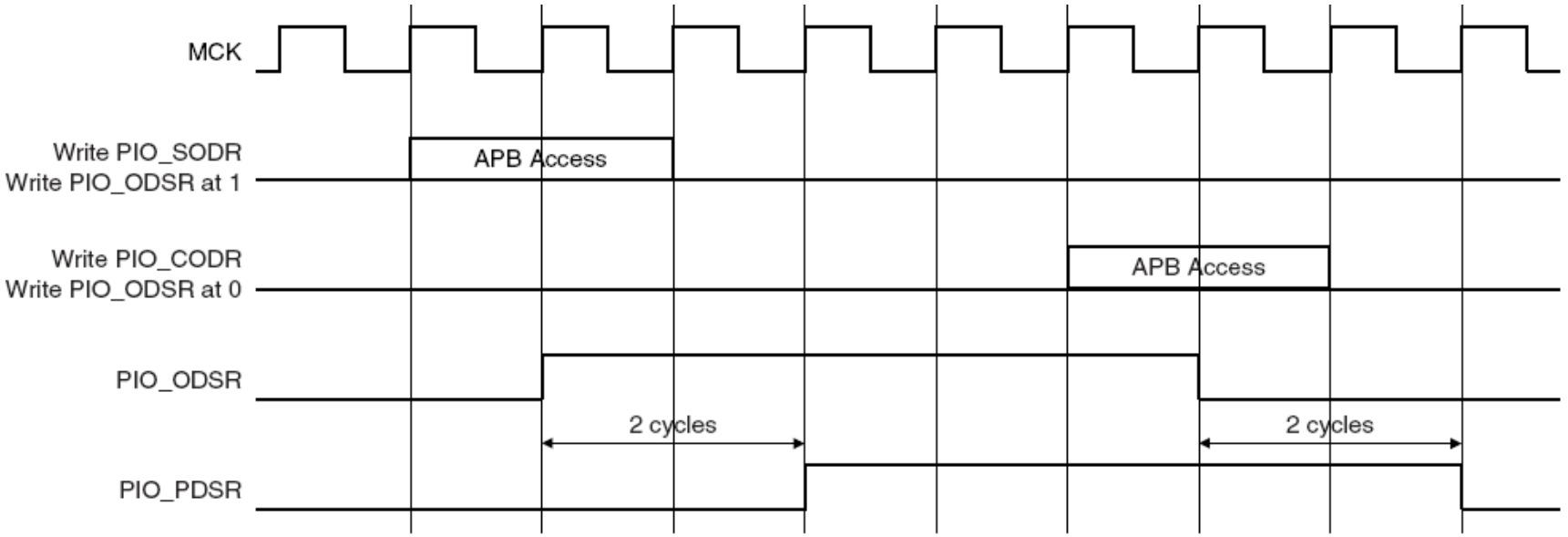
Figure 31-3. I/O Line Control Logic



Schemat blokowy portu I/O – odczyt stanu wejścia



Przebiegi czasowe podczas sterowania portem I/O



- 1 cykl zegarowy opóźnienia, gdy wyjście sterowane jest rejestrami SODR/CODR.
- 2 cykle opóźnienia podczas zapisu całego portu (32 bit, ustwione bity rejestru PIO_OWSR)

Zarządzanie sygnałem zegarowym urządzeń peryferyjnych

PMC Peripheral Clock Enable Register
Register Name:PMC_PCER
Address: 0xFFFFFC10

Table 10-1. AT91SAM9263 Peripheral Identifiers

Peripheral ID	Peripheral Mnemonic	Peripheral Name	External Interrupt
0	AIC	Advanced Interrupt Controller	FIQ
1	SYSC	System Controller Interrupt	
2	PIOA	Parallel I/O Controller A	
3	PIOB	Parallel I/O Controller B	
4	PIOC to PIOE	Parallel I/O Controller C, D and E	
5	reserved		
6	reserved		
7	US0	USART 0	
8	US1	USART 1	
9	US2	USART 2	

```
write_register(PMC_PCER,0x00000110);     // Peripheral clocks 4 and 8 are enabled.
write_register(PMC_PCDR,0x00000010);     // Peripheral clock 4 is disabled.
```

Rejestry odwzorowane w strukturze - powtórzenie

```
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;
        /* 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 ?

```
AT91C_BASE_PIOA->PIO_PER = AT91C_PIO_PA0 | AT91C_PIO_PA19; // ustawia bity 0,19
                                                          // i zeruje pozostałe
```

Definicja rejestrów – pliki nagłówkowe

```
#ifndef _PROJECT_H
#define _PROJECT_H
/*
 * Include your AT91 Library files and specific
 * compiler definitions
 */
#include "AT91SAM9263-EK.h"
#include "AT91SAM9263.h"
#endif  // _PROJECT_H
```

```
/*-----------------------*/
/*  LEDs Definition  */
/*-----------------------*/
#define AT91B_LED1                AT91C_PIO_PB8   /* DS1 */
#define AT91B_LED2                AT91C_PIO_PC29  /* DS2 */
#define AT91B_NB_LEB        2
#define AT91D_BASE_PIO_LED1   (AT91C_BASE_PIOB)
#define AT91D_BASE_PIO_LED2   (AT91C_BASE_PIOC)
#define AT91D_ID_PIO_LED1     (AT91C_ID_PIOB)
#define AT91D_ID_PIO_LED2     (AT91C_ID_PIOC)
```

```
/*-----------------------------*/
/* Push Button Definition  */
/*-----------------------------*/
#define AT91B_BP1             AT91C_PIO_PC5  // Left click
#define AT91B_BP2             AT91C_PIO_PC4  // Right click
#define AT91D_BASE_PIO_BP     AT91C_BASE_PIOC
#define AT91D_ID_PIO_BP       AT91C_ID_PIOCDE
```

Konfiguracja portów I/O procesora

```
#define AT91C_PIO_PB8        (1 <<  8)                    // Pin Controlled by PB8
#define AT91C_BASE_PIOB   (AT91PS_PIO)        0xFFFFF400        // (PIOB) Base Address
```

Konfiguracja Portu w tryb wejścia:

```
/* Enable the periph clock for the PIO controller, This is mandatory when PIO are configured as input */
AT91C_BASE_PMC->PMC_PCER = (1 << AT91C_ID_PIOCDE ); // peripheral clock enable register (port C, D, E)
/* Set the PIO line in input */
AT91C_BASE_PIOD->PIO_ODR = 0x0000.000F;                    // 1 – Set direction of the pin to input
/* Set the PIO controller in PIO mode instead of peripheral mode */
AT91C_BASE_PIOD->PIO_PER = AT91C_PIO_PB8;          // 1 – Enable PIO to control the pin
```

Konfiguracja Portu w tryb wyjścia:

```
/* Configure the pin in output */
AT91C_BASE_PIOB->PIO_OER = AT91C_PIO_PB8 ;
/* Set the PIO controller in PIO mode instead of peripheral mode */
AT91C_BASE_PIOD->PIO_PER = 0xFFFF.FFFF;                    // 1 – Enable PIO to control the pin
AT91C_BASE_PIOE->PIO_PER = AT91C_PIO_PB31;
/* Disable pull-up */
AT91C_BASE_PIOA->PIO_PPUDR = 0xFFFF.0000;                  // 1 – Disable the PIO pull-up resistor
```

Układy do odmierzania czasu - timery procesora

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Timery

Timer – urządzenie peryferyjne procesora przeznaczone do odmierzania określonych przedziałów czasu (zliczania elementarnych cykli zegarowych). Po odmierzeniu wymaganego okresu czasu timer zwykle generuje przerwanie. Timery wykorzystywane są do odmierzania czasu systemowego, przełączania wątków, generacji opóźnień.

Przykładów układów służących do odmierzania czasu:

Timer PIT (ang. Peridic Interval Timer, Programmable Interrupt Timer),

Timer Czasu Rzeczywistego RTT (ang. Real-Time Timer),

Timer PWM (ang. Pulse Width Modulation),

Timer uniwersalny TC (ang. Timer Counter),

Timer Watch-dog WDT.

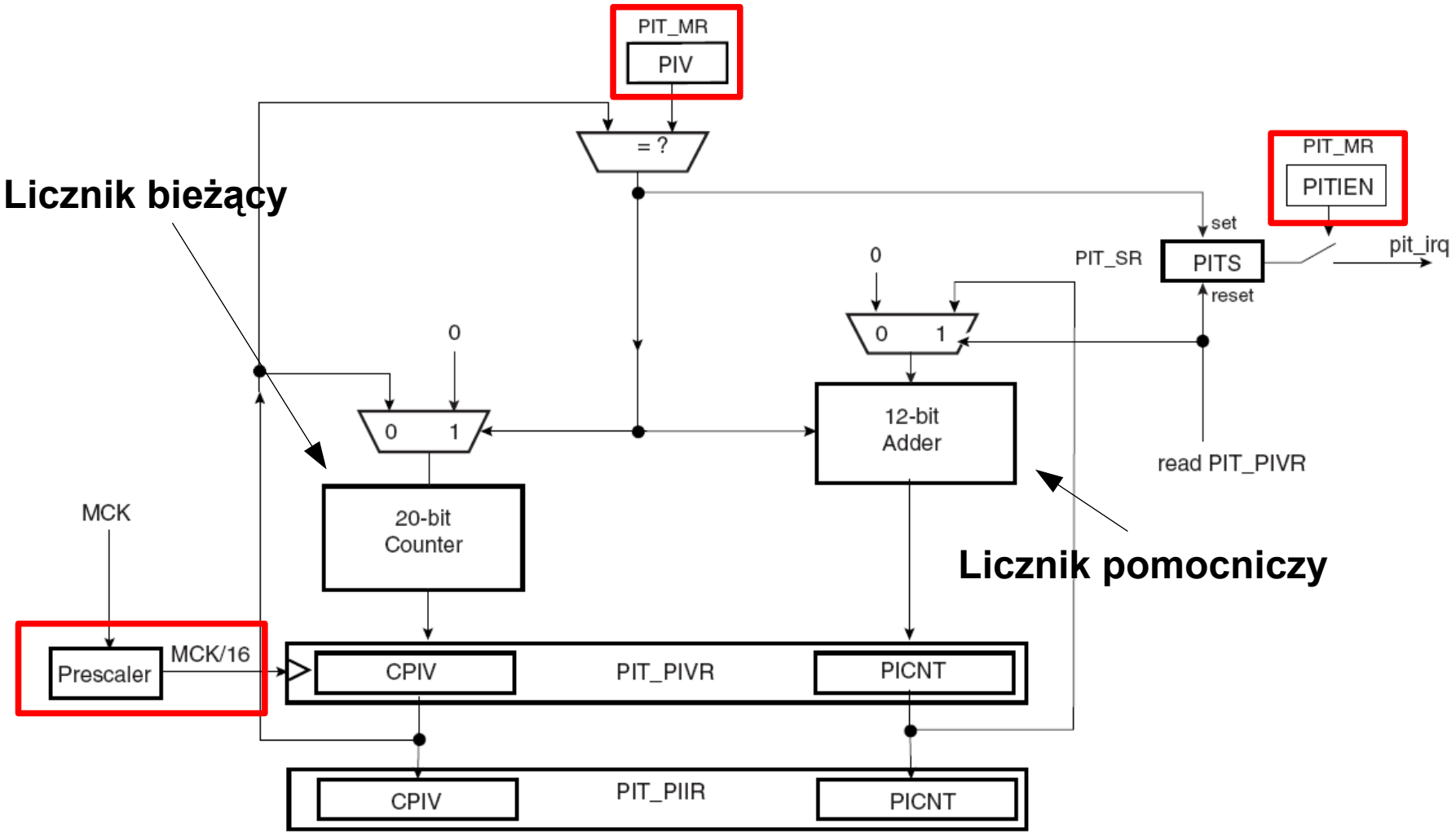
Moduł timera PIT (Periodic Interval Timer)

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

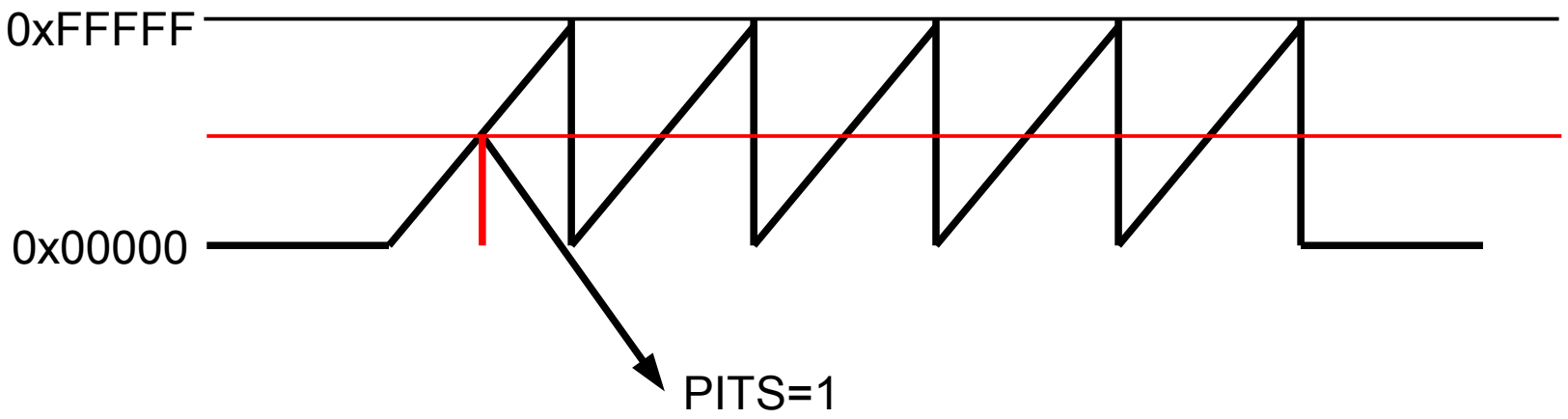
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Schemat blokowy timera PIT



Automatyczne przeładowanie timera

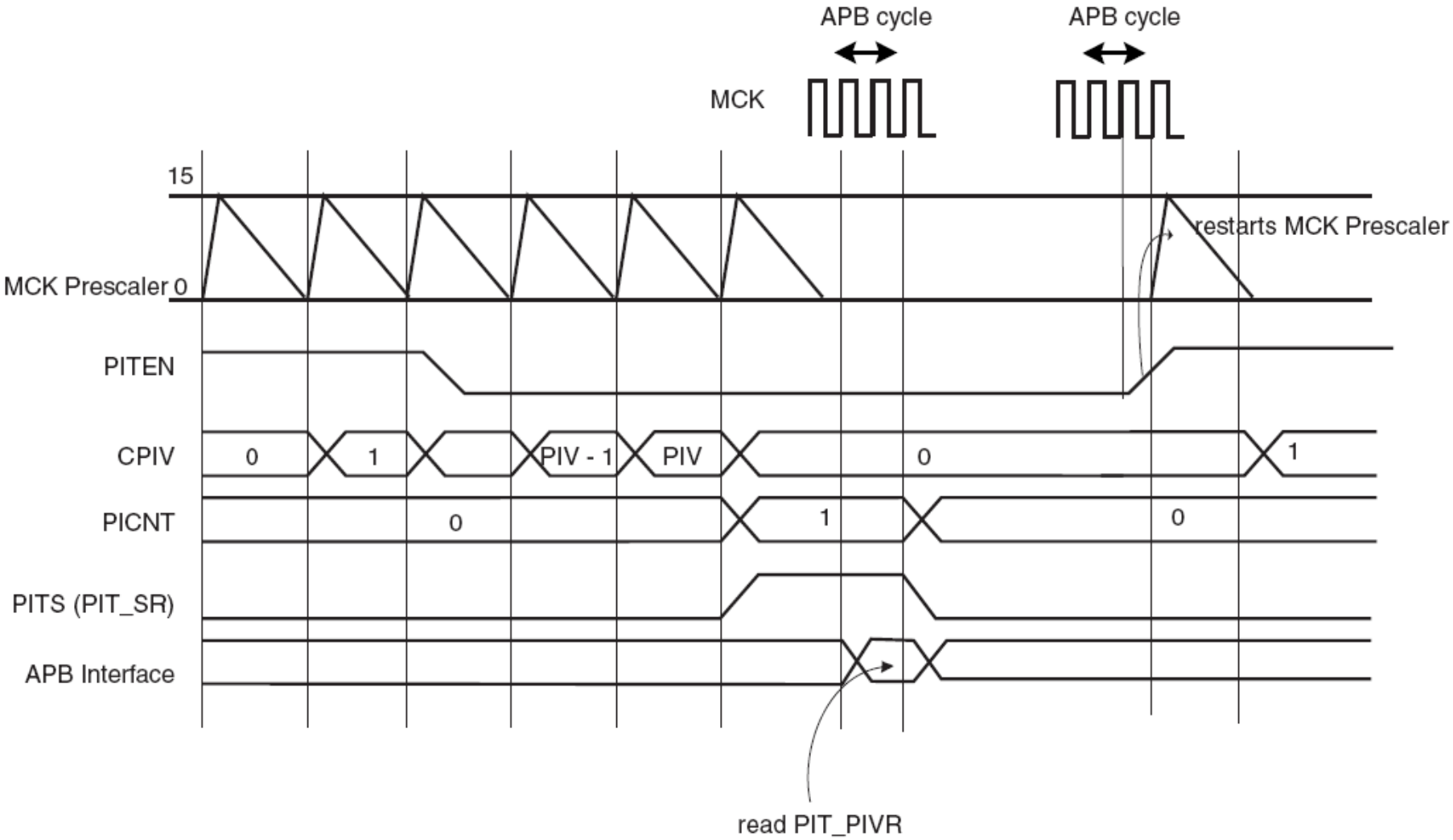


Okres generowanych przerwań:

(PIV_VALUE+1)/(16 * Clk)

Clk = 100 MHz, PIV = 0x1000 => t_{PIT} = 2,5 us

Przebiegi obrazujące pracę timera PIT



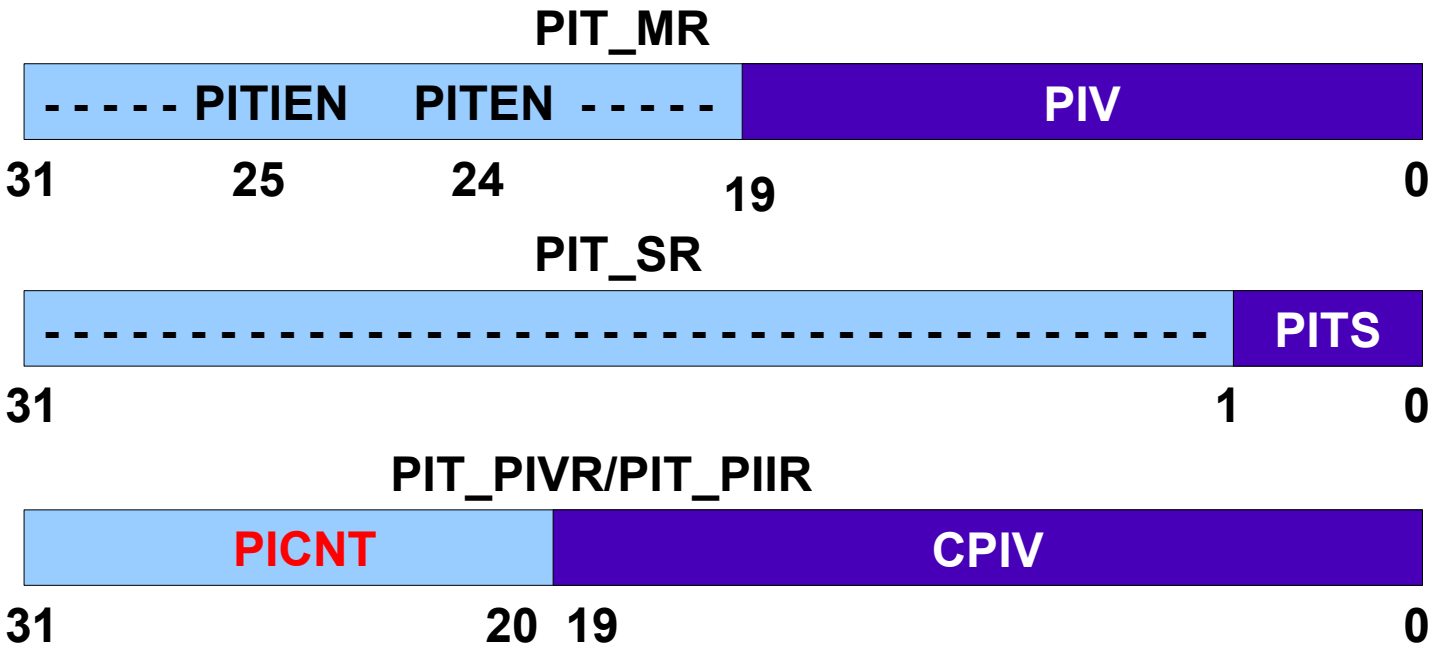
Rejestry timera PIT

Table 16-1. Register Mapping

Offset	Register	Name	Access	Reset
0x00	Mode Register **Address:** 0xFFFFFD30	PIT_MR	Read-write	0x000F_FFFF
0x04	Status Register	PIT_SR	Read-only	0x0000_0000
0x08	Periodic Interval Value Register	PIT_PIVR	Read-only	0x0000_0000
0x0C	Periodic Interval Image Register	PIT_PIIR	Read-only	0x0000_0000

```
typedef struct S_PIT {          /*  Register name                R/W   Reset val.      Offset
AT91_REG    PIT_MR;             // PIT Mode Register             R/W   0x000F.FFFF     0x00
AT91_REG    PIT_SR;             // PIT Status Register           R     0x0000.0000     0x04
AT91_REG    PIT_PIVR;                   // PIT Per. Int. Val. Reg.          R     0x0000.0000
   0x08
AT91_REG    PIT_PIIR;           // PIT Per. Int. Image Reg.      R     0x0000.0000     0x0C
} S_PIT, *PS_PIT;
/* blok rejestrów portów I/O PIOA...PIOE */
#define PIT     ((PS_PIT)    0xFFFFFD30)     // (PIT) Base Address
```

Rejestry timera PIT



Rdzeń procesora ARM

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Architektura ARM (1)

Rdzeń procesora ARM – procesor zgodny z architekturą ARM zaprokjektowany w języku opisu sprzętu (najczęściej VHDL lub Verilog) dostarczony jako makrokomórka (ang. macro-cell) lub IP (ang. Intellectual Property).

Cechy rdzeni procesorów ARM:

- Przeznaczony do dalszej rozbudowy – procesory, SoC
- 32-bitowy procesor zgodny z architekturą RISC
- Zoptymalizowany pod względem niskiego poboru mocy
- Różne tryby pracy:
  - 32-bitowe instrukcje ARM
  - 16-bitowe instrukcje Thumb
  - Instrukcje języka Java - Jazelle DBX
- Praca w trybie Big lub Little Endian
- Szybka obsługa przerwań (FIR – Fast Interrupt Response), aplikacje czasu rzeczywistego
- Pamięć wirtualna
- Lista wydajnych instrukcji (zoptymalizowane na podstawie architektury RISC oraz CISC)
- Sprzętowe wsparcie dla języków wyższego poziomu

Architektura ARM (2)

Nomenklatura:

ARM {x} {y} {z} {T} {D} {M} {I} {E} {J} {F} {S}

- **x** – rodzina rdzenia (nie mylić z numerem architektury)
- **y** – zarządzanie pamięcia/ochrona pamięci (opcjonalnie)
- **z** – pamięć cache (opcjonalnie)
- **T** – Thumb, rdzeń może wykonywać 16-bitowe rozkazy
- **D** – Debug, rdzeń wyposażony w moduł JTAG umożliwiający debugowanie
- **M** – Multiplier, wyposażony w sprzętowy układ mnożący (32x32 = 64 bit)
- **I** – ICE, moduł pozwalający na debugowanie w układzie (ang. In-Circuit Emulator, breakpoints)
- **E –** Enhanced DSP instructions, instrukcje przydatne podczas przetwarzania sygnałów
- **J** – Jazelle,
- **F** – Floating-point, moduł wspomagający obliczenia zmiennoprzecinkowe
- **S** – Synhesizible version, wersja sytezowalna – dostępne kody źródłowe dla narzędzi EDA

Przykłady:

ARM7TDMI **ARM926-EJ-S**

Architektura ARM (3)

- **Wersja 1, v1**
  - podstawowe operacje arytmetyczno/logiczne,
  - programowe przerwania,
  - 8 i 32 bitowe operacje na danych,
  - 26-bitowy adres
- **Wersja 2, v2**
  - jednostka mnożąca oraz mnóż i sumuj (MAC),
  - dostępny koprocesor,
  - opercje służace do synchronizacji wątków,
  - 26-bitowy adres
- **Wersja 3, v3 (AMR 6-7)**
  - nowe rejestry CPSR, SPSR, MRS, MSR,
  - dostępne dostatkowe tryby pracy Abort i Undef,
  - 32-bitowy adres

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Architektura ARM (4)

- **Wersja 4, v4 (ARM <=7, np. ARM7TDMI-S → Philips LPC2000 family)**
  - pierwsza oficjalnie zatwierdzona architektura
  - w prowadzono 16-bitowe operacje na danych
  - tryb pracy THUMB
  - dodatkowy uprzywilejowany tryb pracy procesora (supervisor mode)
  - możliwość inkrementacji PC o podwójne słowo (64 bit)
- **Wersja 5, v5 (AMR7-9, np. ARM926EJ-S → ATMEL AT91SAM9263)**
  - ulepszona współpraca procesora pomiędzy trybami ARM/THUMB, możliwość zmiany trybu procesora w czasie wykonywania programu
  - dodana instrukcja CLZ (Count Leading Zeros)
  - możliwość użycia programowych pułapek
  - wsparcie dla pracy wieloprocesorowej
- **Wersja 6, v6 (ARM11)**
  - ulepszona jednostka zarządzania pamięcia MMU (Management Memory Unit)
  - sprzętowe wsparcie dla przetwarzania dźwięku i obrazu (FFT, MPEG4, SIMD, etc...)
  - ulepszona obsługa wyjątków (nowa flaga w rejestrze CPSR)

Porównanie wybranych procesorów ARM

Family	Architecture Version	Core	Feature	Cache (I/D)/MMU	Typical MIPS @ MHz
ARM6	ARMv3	ARM610	Cache, no coprocessor	4K unified	17 MIPS @ 20 MHz
ARM7	ARMv3	ARM7500FE	Integrated SoC. "FE" Added FPA and EDO memory controller.	4 KB unified	55 MIPS @ 56 MHz
ARM7TDMI	ARMv5TEJ	ARM7EJ-S	Jazelle DBX, Enhanced DSP instructions, 5-stage pipeline	8 KB	120 MIPS @ 133 MHz
StrongARM	ARMv4	SA-110	5-stage pipeline, MMU	16 KB/16 KB, MMU	235 MIPS @ 206 MHz
ARM8	ARMv4	ARM810[7]	5-stage pipeline, static branch prediction, double-bandwidth memory	8 KB unified, MMU	1.0 DMIPS/MHz
ARM9TDMI	ARMv4T	ARM920T	5-stage pipeline	16 KB/16 KB, MMU	245 MIPS @ 250 MHz
ARM9E	ARMv5TEJ	ARM926EJ-S	Jazelle DBX, Enhanced DSP instructions	variable, TCMs, MMU	220 MIPS @ 200 MHz
ARM10E	ARMv5TE	ARM1020E	VFP, 6-stage pipeline, Enhanced DSP instructions	32 KB/32 KB, MMU	300 MIPS @ 325 MHz
XScale	ARMv5TE	PXA27x	MMX and SSE instruction set, four MACs,	32 Kb/32 Kb, MMU	800 MIPS @ 624 MHz
ARM11	ARMv6	ARM1136J(F)-S	SIMD, Jazelle DBX, VFP, 8-stage pipeline	variable, MMU	740 @ 532-665 MHz
Cortex	ARMv7-A	Cortex-A8	Application profile, VFP, NEON, Jazelle RCT, Thumb-2, 13-stage superscalar pipeline	variable (L1+L2), MMU+TrustZone	>1000 MIPS@ 600 M-1 GHz

Rozkazy procesora ARM

Z punktu widzenia wykonywanych instrukcji procesor ARM może pracować w jednym z następujących trybów:

- **ARM** – definiuje 32-bitowe instrukcje (kod programu musi być wyrównany do granicy 4 bajtów),
- **Thumb**, **Thumb-2** – definiuje zestaw 16-bitowych instrukcji zoptymalizowanych pod kątem (kod programu musi być wyrównany do granicy 2 bajtów, wszystkie rejestry pracują w trybie 32 bitowym),
- **Jazelle v1** – tryb pozwalający na bezpośrednie wykonywanie instrakcji zgodnych ze specyfikacją języka Java (bez użycia maszyny wirtualnej JVM, 1000 Caffeine Marks @ 200MHz)

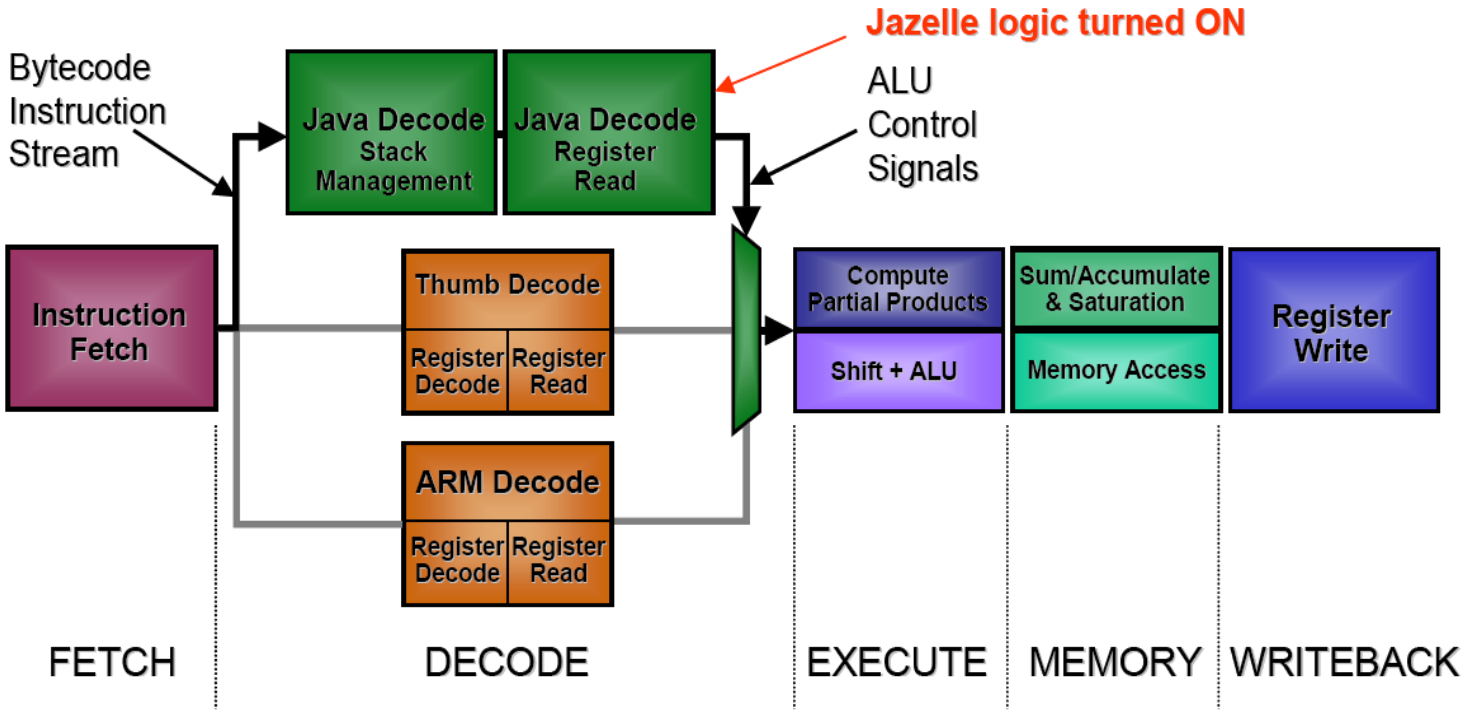
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Wsparcie dla języka Java

- Oznaczenie rdzenia procesora z literą 'J'
- Dynamiczne podmiana rejestrów oraz stosu
- Sprzętowa realizacja dekodera instrukcji – nie maszyna wirtualna



Model programowy – rejestry procesora

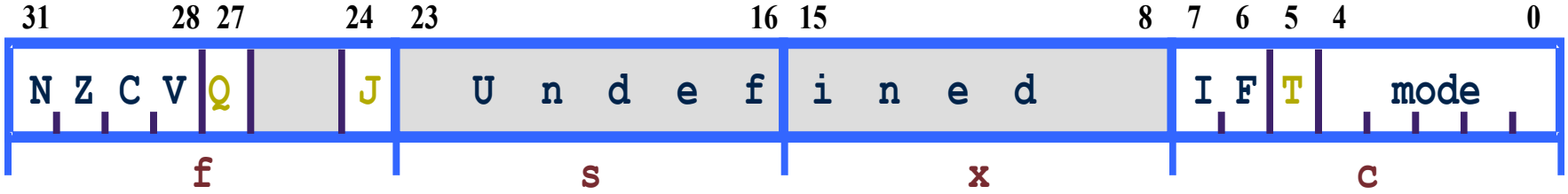
Procesor ARM posiada łącznie 37 rejestrów (wszystkie są 32 bitowe):

- **PC (r15)** – licznik programu (Program Counter)
- **CPSR** – rejestr statusowy, obecny status (Current Program Status Register)
- **SPSR** – rejestr statusowy, dostępne w różnych trybach uprzywilejowania (Saved Program Status Register)
- **LR (r14)** – rejestr powrotu (Link Register), wykorzystywany podczas wywoływania fukcji (zapamiętuje PC – adres powrotu)
- **SP (r13)** – zwykle używany jako wskaźnik stosu (Stack Pointer)
- **r0 - r12** – rejestry ogólnego przeznaczenia

Uwaga :

- Nie wszystkie rejestry są dostępne w różnych trybach uprzywilejowania procesora

Rejestr statusowy SPSR



Wskaźniki stanu

- V – przepełnie podaczas operacji ALU (oVerflow)
- C – przeniesienie/pożyczka podczas operacji ALU
- Z – ujemny wynik podczas operacji ALU
- N – ujemny wynik operacji ALU lub „mniejszy niż”

Wskaźniki dostępne jedynie dla architektury ARM v5TE/J

- J – Procesor w trybie Jazelle
- Q – Sticky Overflow – wskaźnik nasycenia podczas operacji ALU (QADD, QDADD, QSUB or QDSUB, lub rezultat operacji SMLAxy or SMLAWx przekracza 32-bity)

Przerwania

- I=1 Przerwania IRQ wyłączone
- F=1 Przerwania FIQ wyłączone

Wskaźniki dostępne dla arch. xT

- T=1 Tryb pracy ARM
- T=1 Tryb pracy Thumb

Tryb pracy procesora

- Definiują jeden z 7 trybów

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

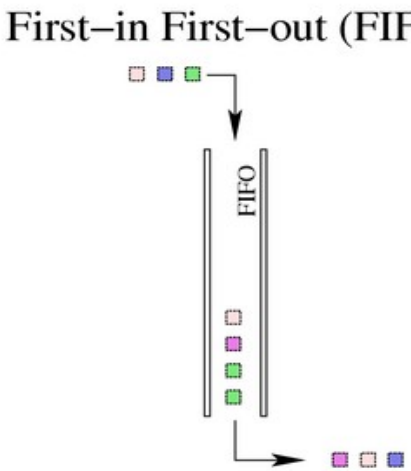
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

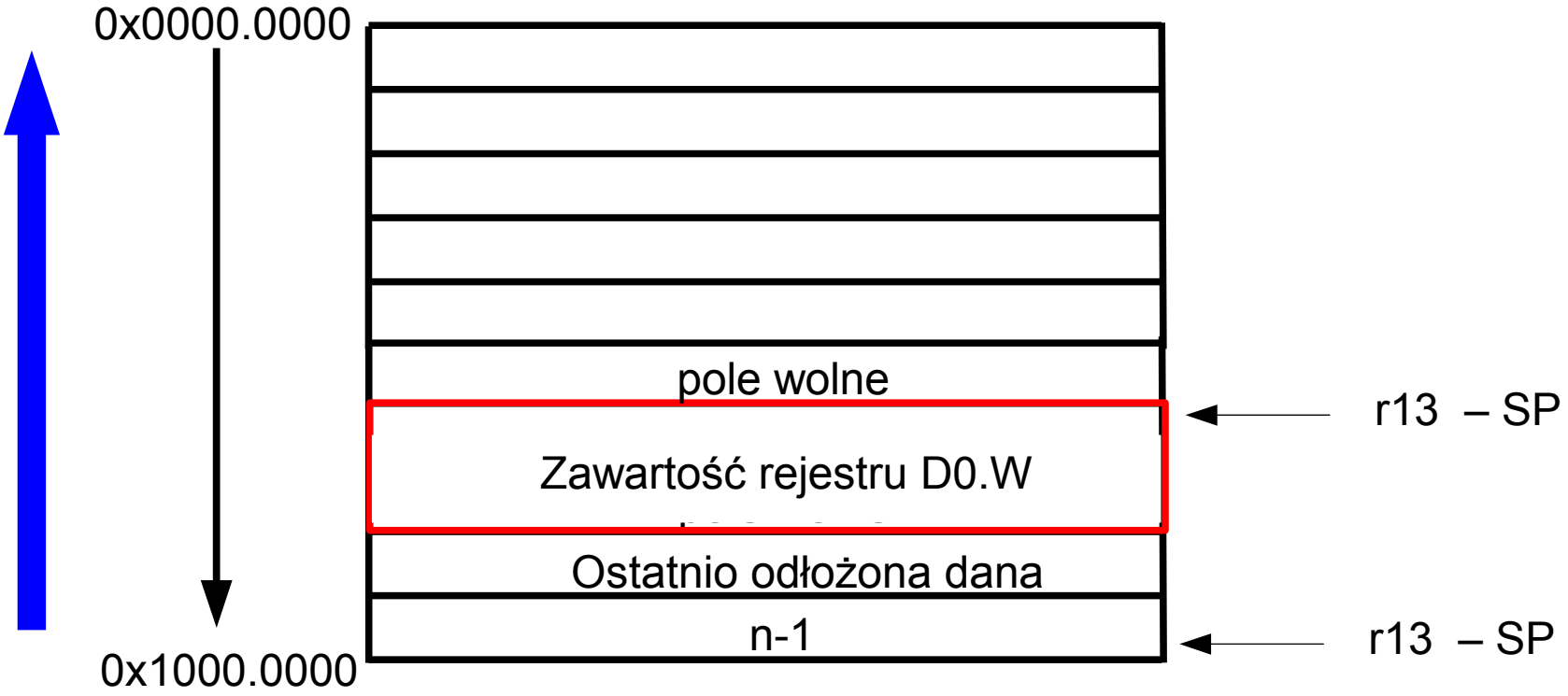
Struktura stosu (1)

Stos (ang. stack lub LIFO Last-In, First-Out) - liniowa struktura danych, w której dane odkładane są na wierzch stosu i z wierzchołka stosu są zdejmowane. Ideę stosu danych można zilustrować jako stos położonych jedna na drugiej książek – nowy egzemplarz kładzie się na wierzch stosu i z wierzchu stosu zdejmuje się kolejne egzemplarze. Elementy stosu poniżej wierzchołka stosu można wyłącznie obejrzeć, aby je ściągnąć, trzeba najpierw po kolei ściągnąć to, co jest nad nimi

FIFO (ang. First In, First Out) - przeciwieństwem stosu LIFO jest kolejka, bufor typu FIFO (pierwszy na wejściu, pierwszy na wyjściu), w którym dane obsługiwane są w takiej kolejności, w jakiej zostały dostarczone (jak w kolejce do kasy)



Struktura stosu (2)

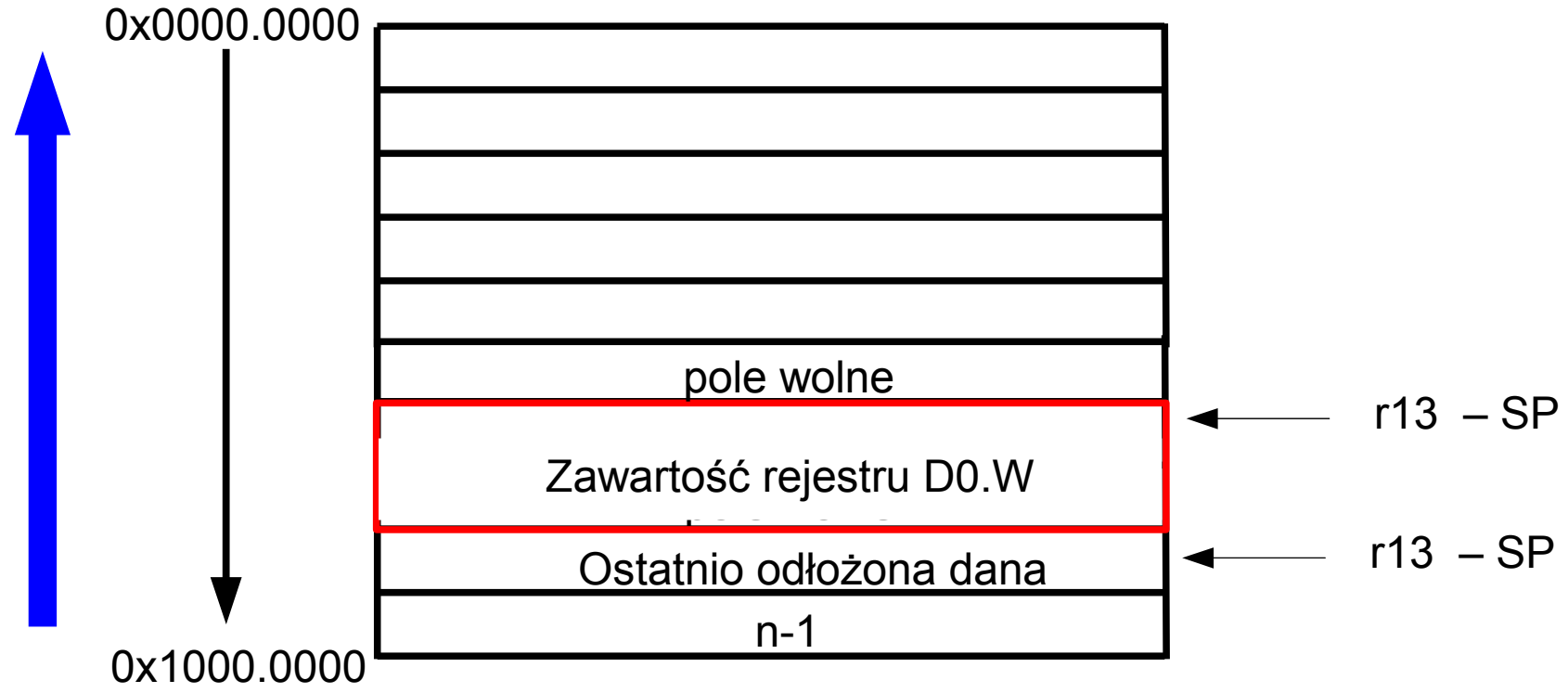


Rejestr r13 jest wskaźnikiem stosu

```
MOVE.W      (r13)+, %r0     | zdjęcie słowa ze stosu i zapisanie r0.W, po przesłaniu danej
                            | rejestr A7 jest zwiększany o 2
```

Struktura stosu (3)

A7 - wskaźnik stosu



MOVE.W %r0, -(r13) | zmniejszenie A7 o 2, odłożenie zawartości r0.W na stos,

Model programowy – dostępne tryby pracy procesora (1)

Tryb pracy procesora (operating mode), tryb ochrony - określa jakie zasoby procesora są dostępne, np. dostępne rejestry, obszary pamięci, urządzenia peryferyjne.

Procesory ARM mogą pracować w jednym z siedmiu trybów pracy:

- **User –** tryb użytkownika (nieuprzywilejowany) przeznaczony do wykonywania programów użytkownika, brak dostępu blokowania przerwań
- **FIQ –** tryb obsługujący przerwania i wyjątki o wysokich priorytetach (szybki) – wykorzystywany do obsługi czasowo krytycznych przerwań (operacje czasu rzeczywistego)
- **IRQ –** obsługa przerwań z niskim prirytetem (low/normal priority)
- **Supervisor** – tryb pracy superużytkownika, dostęp do wszystkich zasobów procesora *(dostępny po resecie lub przerwanie programowe),* przerwania programowe (instrukcja SWI)
- **Abort –** obsługa wyjątków związanych z pamięcią (memory access violations)
- **Undef (undifined mode)–** obsługa nieznanych/błędnych rozkazów
- **System –** tryb pracy superużytkownika, dostęp do rejestrów takich jak w trybie User jednak możliwy dostęp do różnych obszarów pamięci – wykorzystywany przez system operacyjny

Model programowy – dostępne tryby pracy procesora (2)

Mode	Abbreviation	Privileged	Mode[4:0]
Abort	abt	yes	10111
Fast interrupt request	fiq	yes	10001
Interrupt request	irq	yes	10010
Supervisor	svc	yes	10011
System	sys	yes	11111
Undefined	und	yes	11011
User	usr	no	10000

W procesorach ARM zastosowano mechanizm bankowania rejestrów. Zmiana trybu pracy procesora powoduje podmienienie części rejestrów. Pozwala to na implementację oddzielnego stosu dla każdego trybu pracy, co znacząco podnosi stabilność systemu. Zmiana trybu pracy odbywa się automatycznie lub przez modyfikację rejestru CPSR (pole Mode).

Model programowy – rejestry dostępne w trybie User oraz System

Current Visible Registers

User Mode

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (sp)
r14 (lr)
r15 (pc)

cpsr

Banked out Registers

FIQ	IRQ	SVC	Undef	Abort
r8				
r9				
r10				
r11				
r12				
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)
spsr	spsr	spsr	spsr	spsr

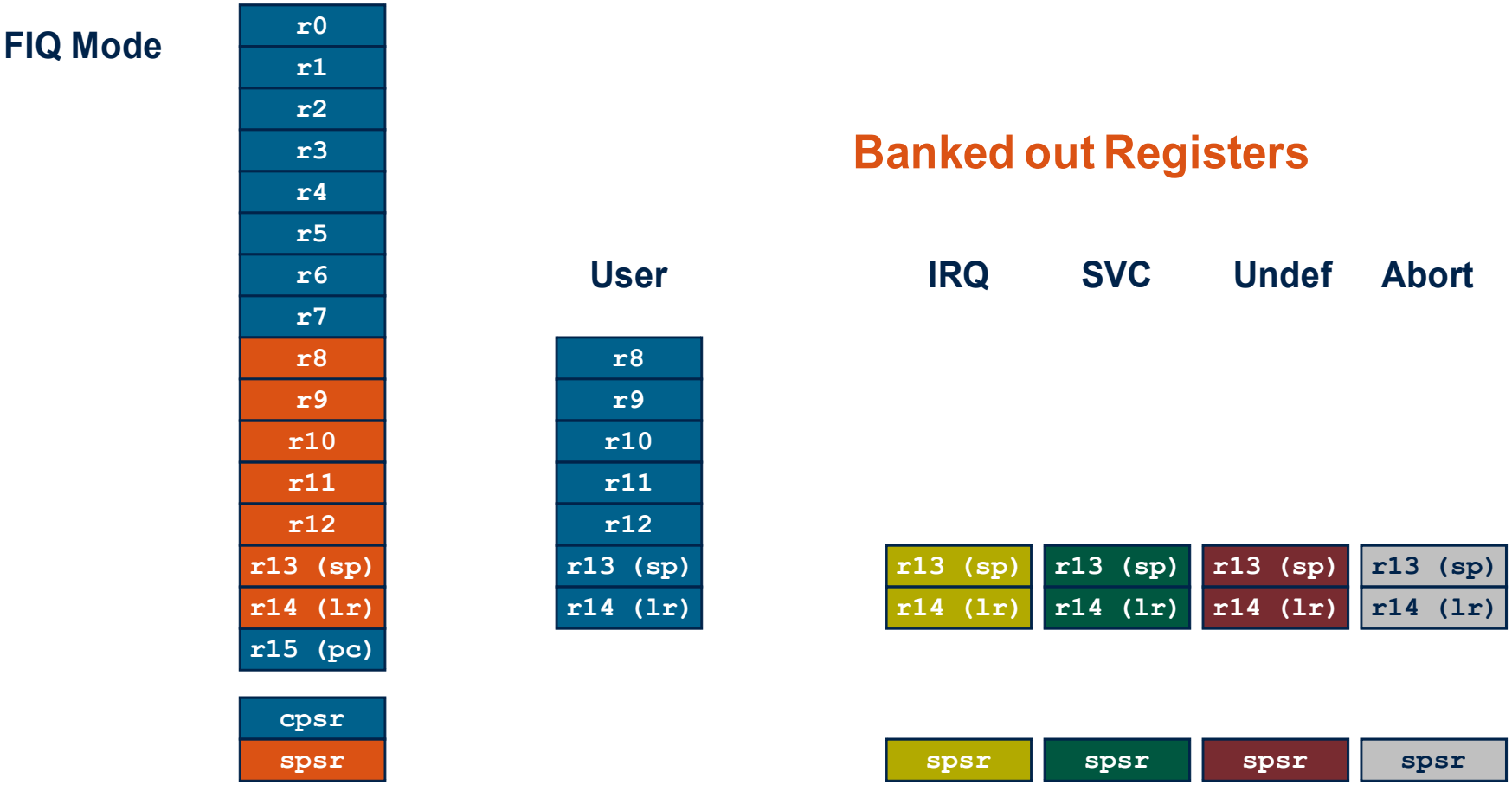
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

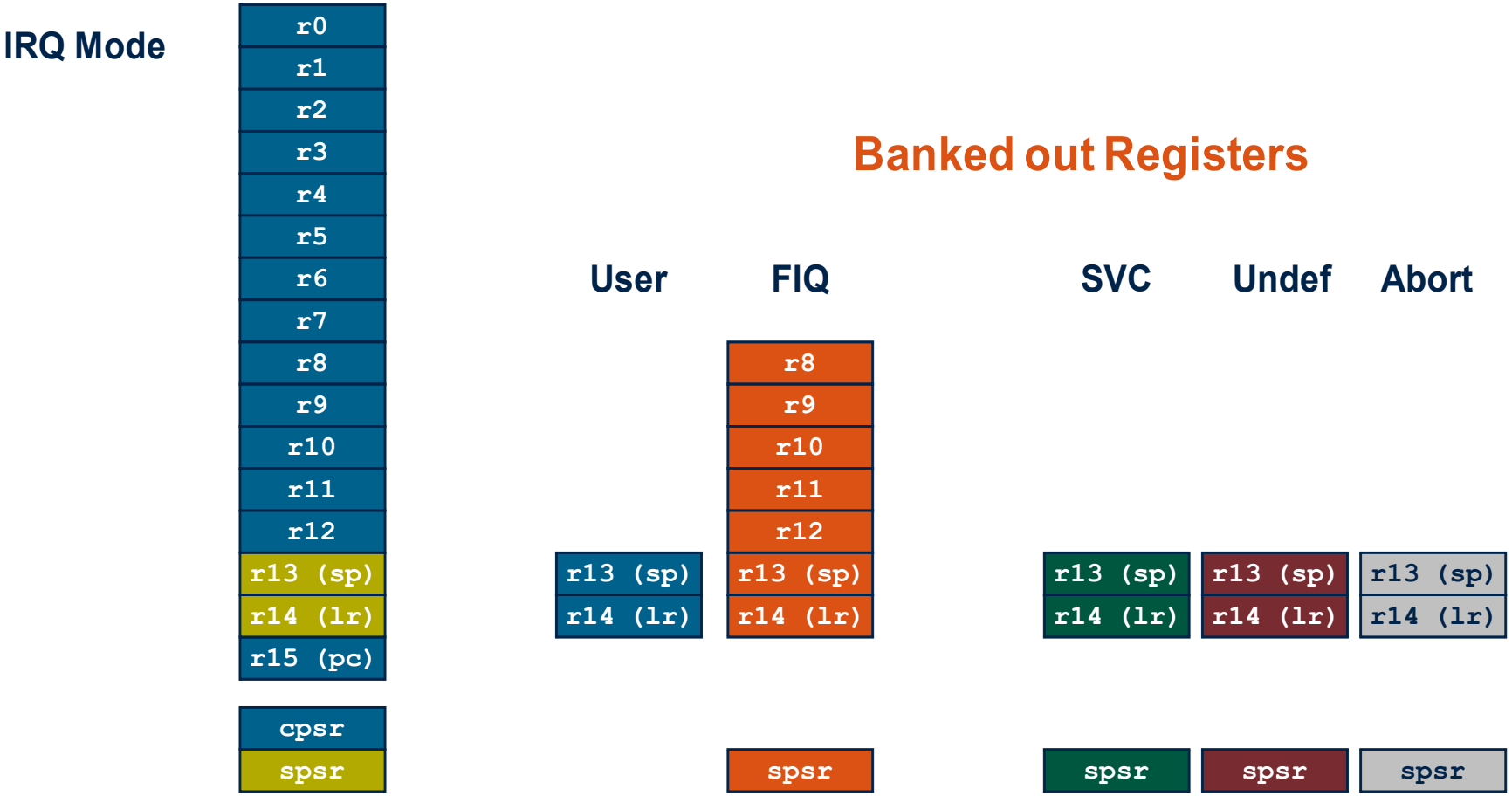
Model programowy – rejestry dostępne w trybie FIQ

Current Visible Registers



Model programowy – rejestry dostępne w trybie IRQ

Current Visible Registers



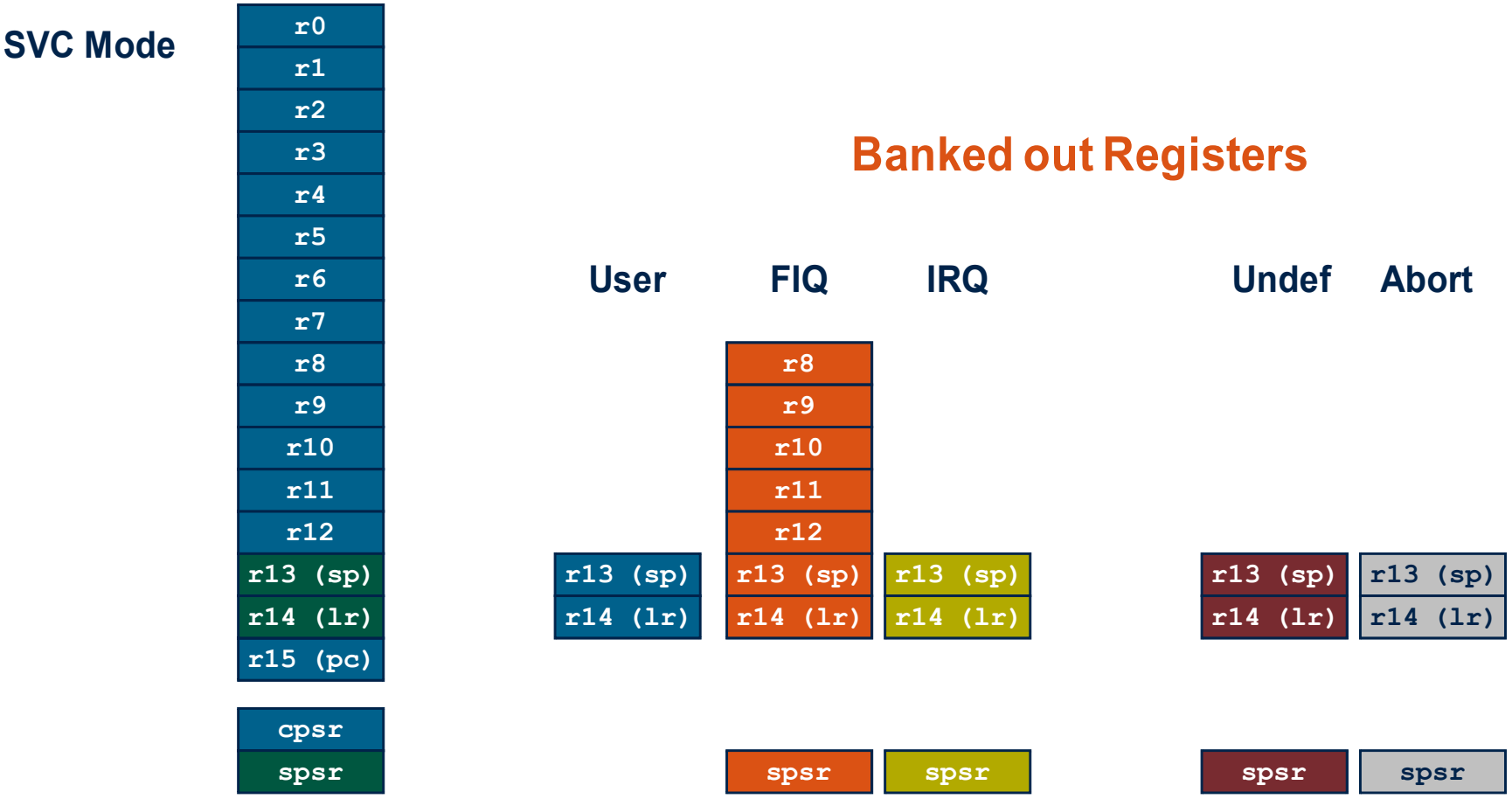
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

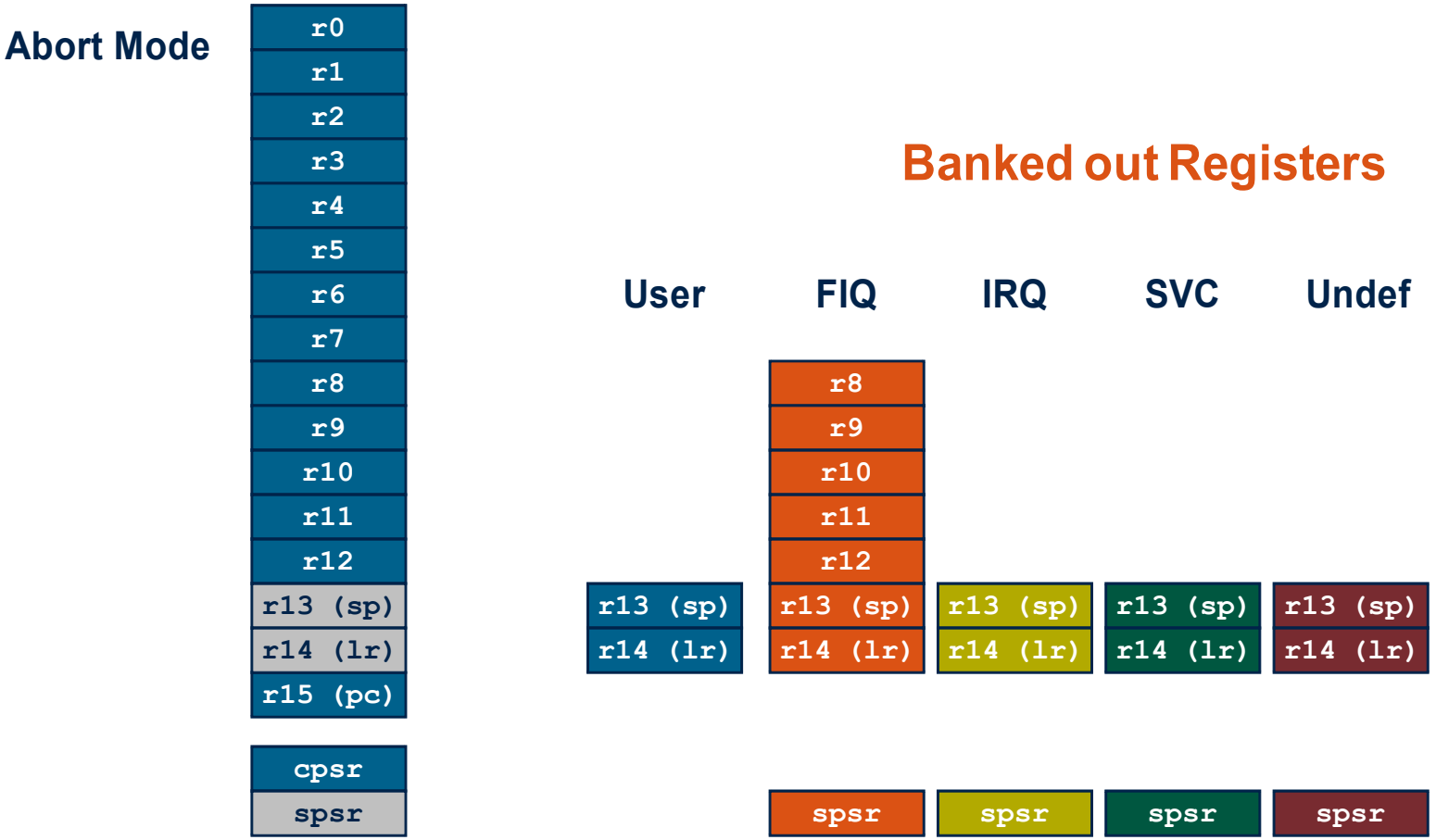
Model programowy – rejestry dostępne w trybie Supervisor

Current Visible Registers



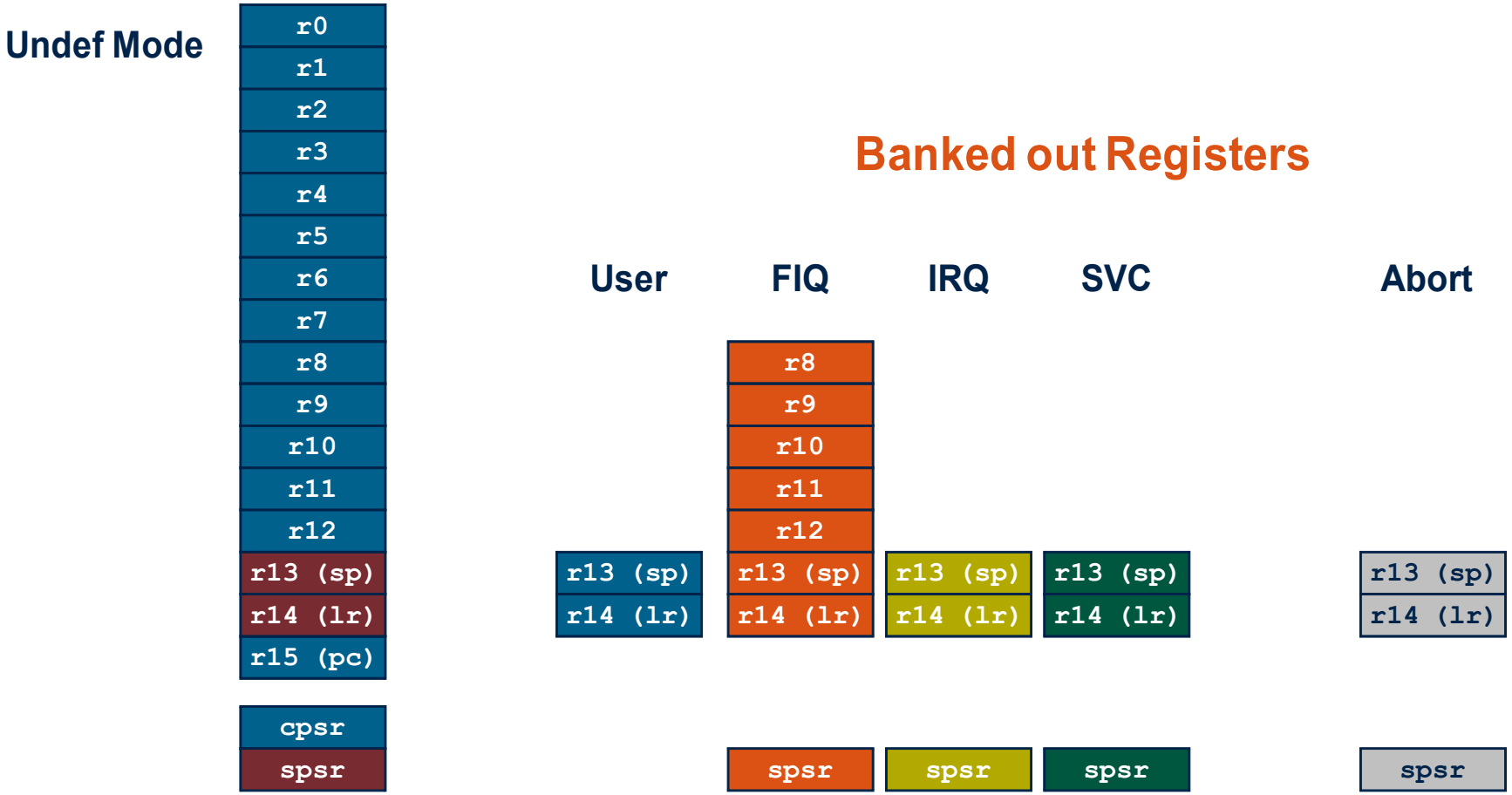
Model programowy – rejestry dostępne w trybie Abort

Current Visible Registers

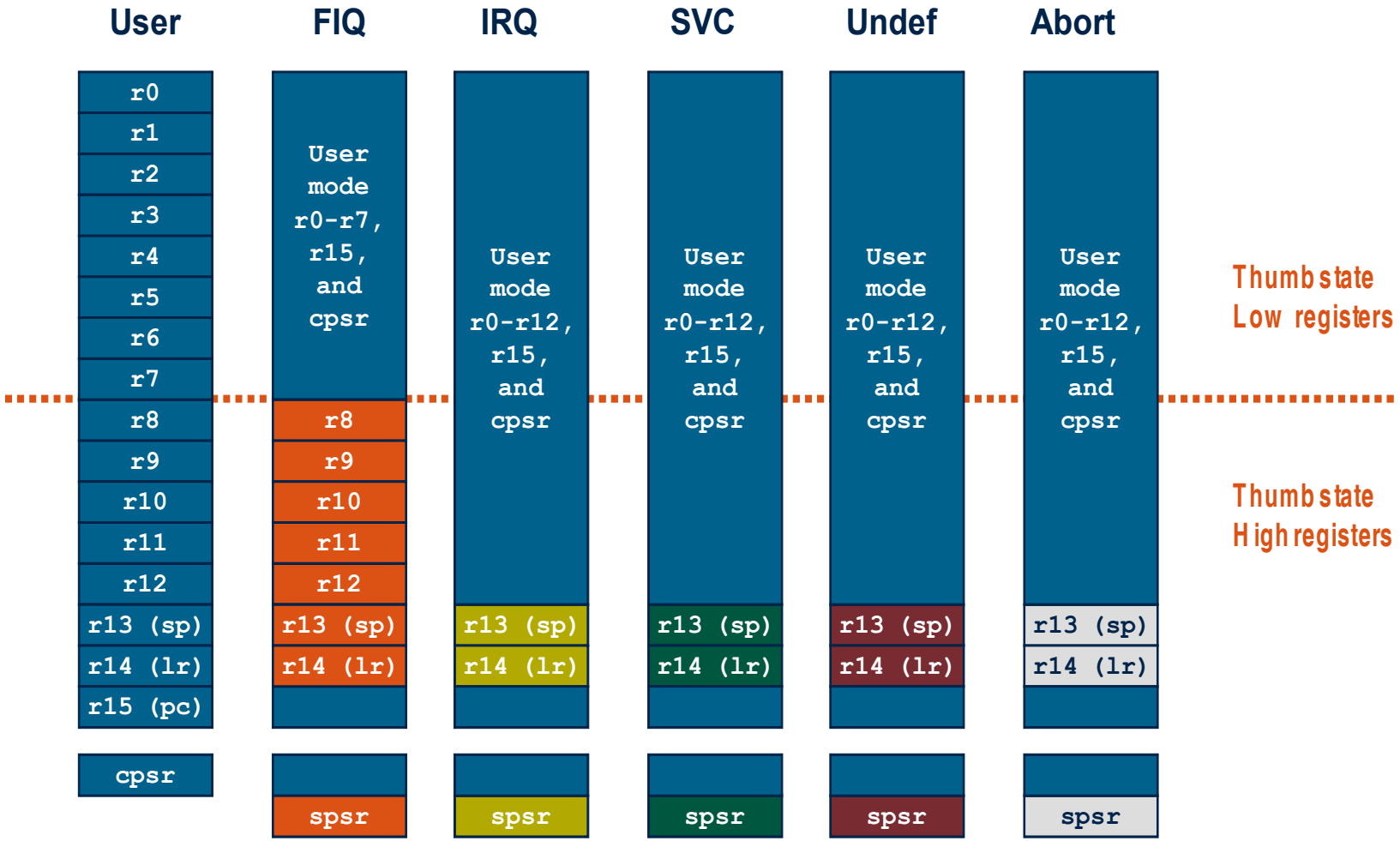


Model programowy – rejestry dostępne w trybie Undef

Current Visible Registers



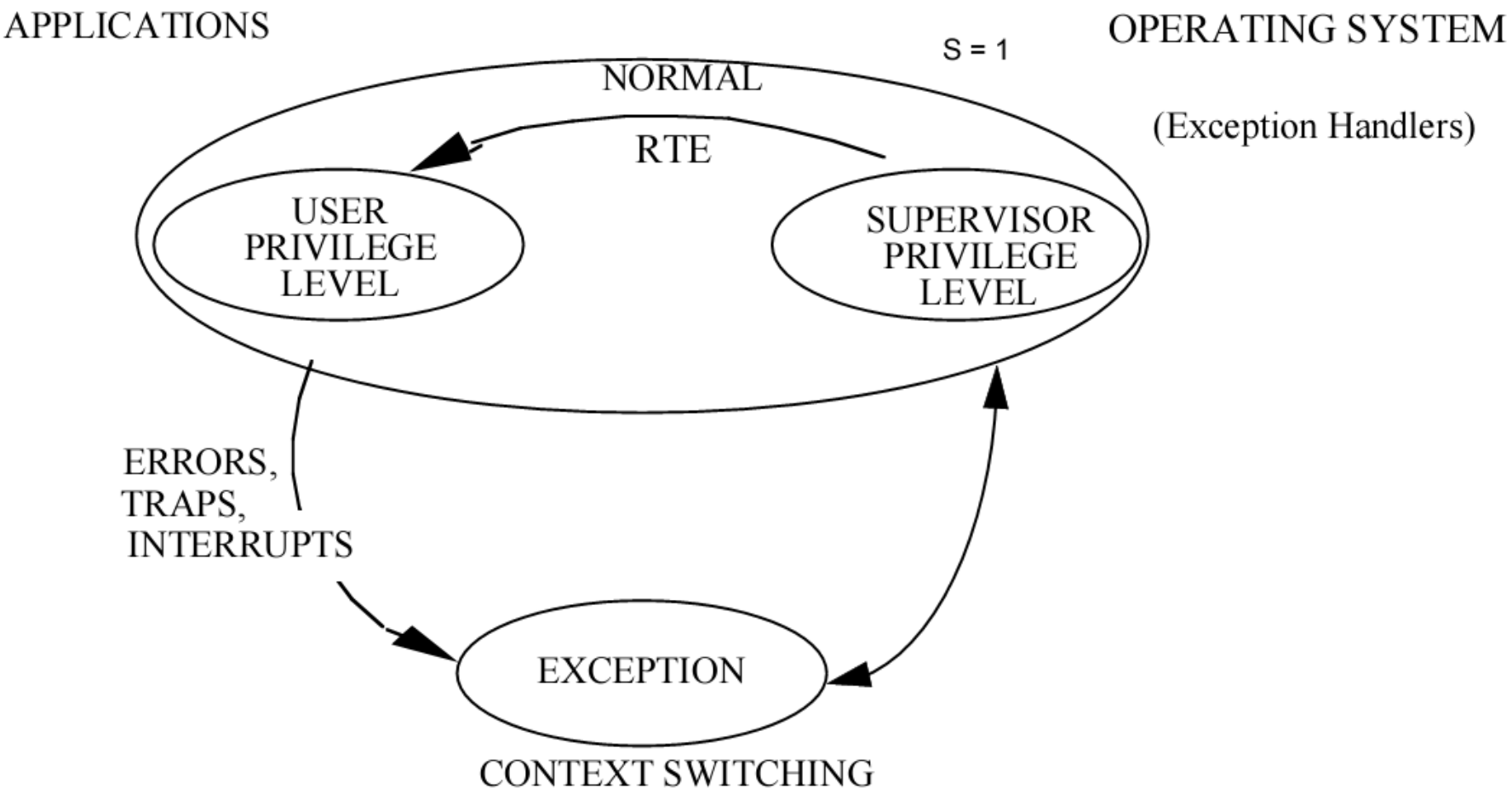
Model programowy – podsumowanie dostępnych rejestrów



Note: System mode uses the User mode register set

Kontroler przerwań AIC (Advanced Interrupt Controller)

Obsługa sytuacji wyjątkowych



Wyjątki

Wyjątek (ang. exception) – mechanizm kontroli przepływu danych występujący w mikroprocesorach oraz we współczesnych językach programowania służący do obsługi zdarzeń wyjątkowych, a w szczególności sytuacji błędnych. Szczególnym przypadkiem wyjątku jest przerwanie.

Wyjątki dzielimy na:

- przerwania (ang. interrupts),
- niepowodzenia (ang. fault),
- błędy nienaprawialne (ang. abort),
- pułapki (ang. trap).

Procesory ARM obsługują dwa rodzaje przerwań:

- **FIQ** (Fast interrupt) – przerwania z szybką obsługą,
- **IRQ** (Interrupt) – przerwania normalne.

Przerwania

Przerwanie (ang. interrupt) lub żądanie przerwania (IRQ – Interrupt ReQuest) – sygnał powodujący zmianę przepływu sterowania, niezależnie od aktualnie wykonywanego programu. Pojawienie się przerwania powoduje wstrzymanie aktualnie wykonywanego programu i wykonanie przez procesor kodu procedury obsługi przerwania, uchwytu przerwania (ang. interrupt handler).

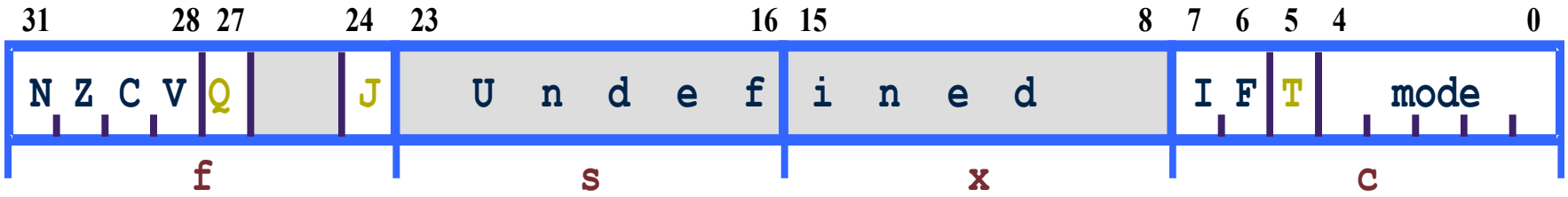
W procesorach ARM wystąpienie przerwania powoduje ustawienie **flagi w rejestrze statusowym** sygnalizującej żądanie obsługi przerwania. Jeżeli system przerwań jest aktywny (rdzeń procesora) oraz dane przerwanie nie jest **zamaskowane** (sterownik przerwań) następuje przyjęcie przerwania przez procesor - skok do programu obsługującego przerwanie.

Przykłady przerwań:

- odebranie lub zakończenie transmisji danej przez port szeregowy,
- zmiana stanu wyprowadzenia portu procesora.

Stan urządzenia można sprawdzać programowo, jednak wymaga to ciągłego sprawdzania stanu rejestru statusowego. Taka operacja nazywana jest odpytywaniem (ang. poolling). Powoduje to znaczne obciążenie procesora, np. transmisja jednego znaku zajmuje ok. 100 us (procesor może w tym czasie wykonać kilka tysiące operacji).

Rejestr statusowy SPSR procesora ARM



Wskaźniki stanu

- V – przepełnie podczas operacji ALU (oVerflow)
- C – przeniesienie/pożyczka podczas operacji ALU
- Z – ujemny wynik podczas operacji ALU
- N – ujemny wynik operacji ALU lub „mniejszy niż”

Wskaźniki dostępne jedynie dla architektury 5TE/J

- J – Procesor w trybie Jazelle
- Q – Sticky Overflow – wskaźnik nasycenia podczas operacji ALU (QADD, QDADD, QSUB or QDSUB, lub rezultat operacji SMLAxy or SMLAWx przekracza 32-bity)

Przerwania

- I=1 Przerwania IRQ wyłączone
- F=1 Przerwania FIQ wyłączone

Wskaźniki dostępne dla arch. xT

- T=0 Tryb pracy ARM
- T=1 Tryb pracy Thumb

Tryb pracy procesora

- Definiują jeden z 7 trybów operacyjnych rdzenia procesora

Obsługa sytuacji wyjątkowych

Wykonanie niedozwolonej operacji przez procesor w danym stanie uprzywilejowania powoduje wygenerowanie wyjątku.

Obsługa wyjątku obejmuje wszystkie operacje od momentu wykrycia błędu do pobrania pierwszej instrukcji obsługującej sytuację wyjątkową.

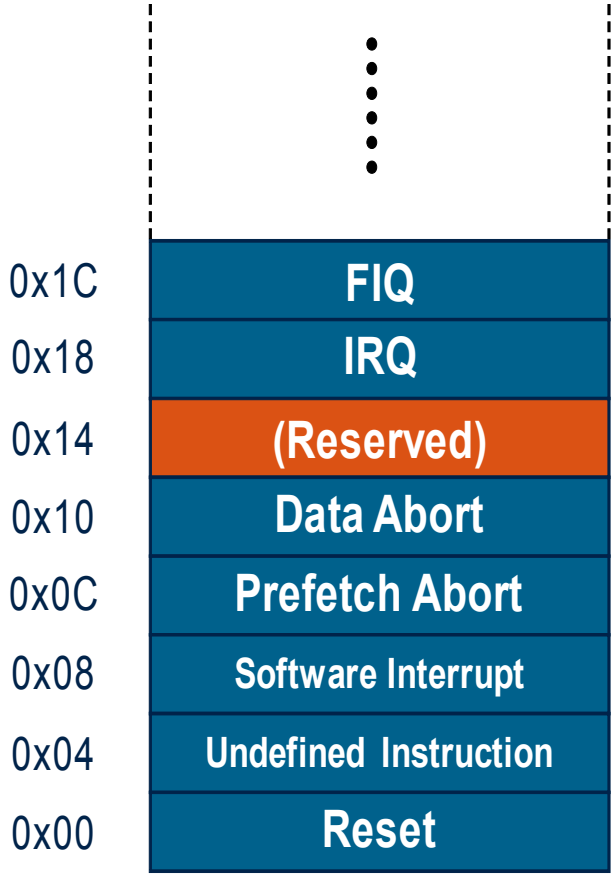
1. a) Wykonanie kopii CPSR → SPSR oraz PC (r15) → Link Register (r14),
   b) Przejście do trybu ARM (z trybu Thumb lub Jazelle),
   c) Przejście do trybu obsługi przerwań (FIQ/IRQ) lub wyjątków,
   d) Ustawienie maski IRQ na poziomie zgłaszanego przerwania (lub wyłączenie przerwań).
   e) Przełączenie banku rejestrów,
   f) Uaktywnienie rejestru SPSR.
2. Określenie wektora obsługiwanego wyjątku (przerwania).
3. Obliczenie adresu pierwszej instrukcji procedury obsługującej dany wyjątek (przerwanie).

Powrót z obsługi sytuacji wyjątkowych

1. a) Odtworzenie rejestru CPSR (r15),
   b) PC (Link Register r14),
   c) Powrót do wykonywanego rozkazu.

Tablica wektorów przerwań umieszczona po resecie pod adresem 0x0.

Tablice można przesunąć pod adres 0xFFFF.0000 (ARM 7/9/10).



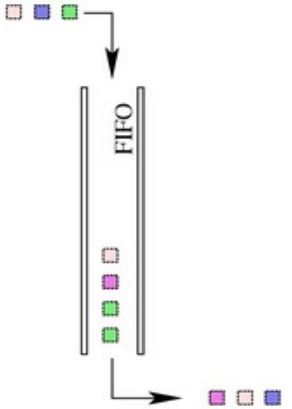
Fragment pamięci

Struktura stosu (1)

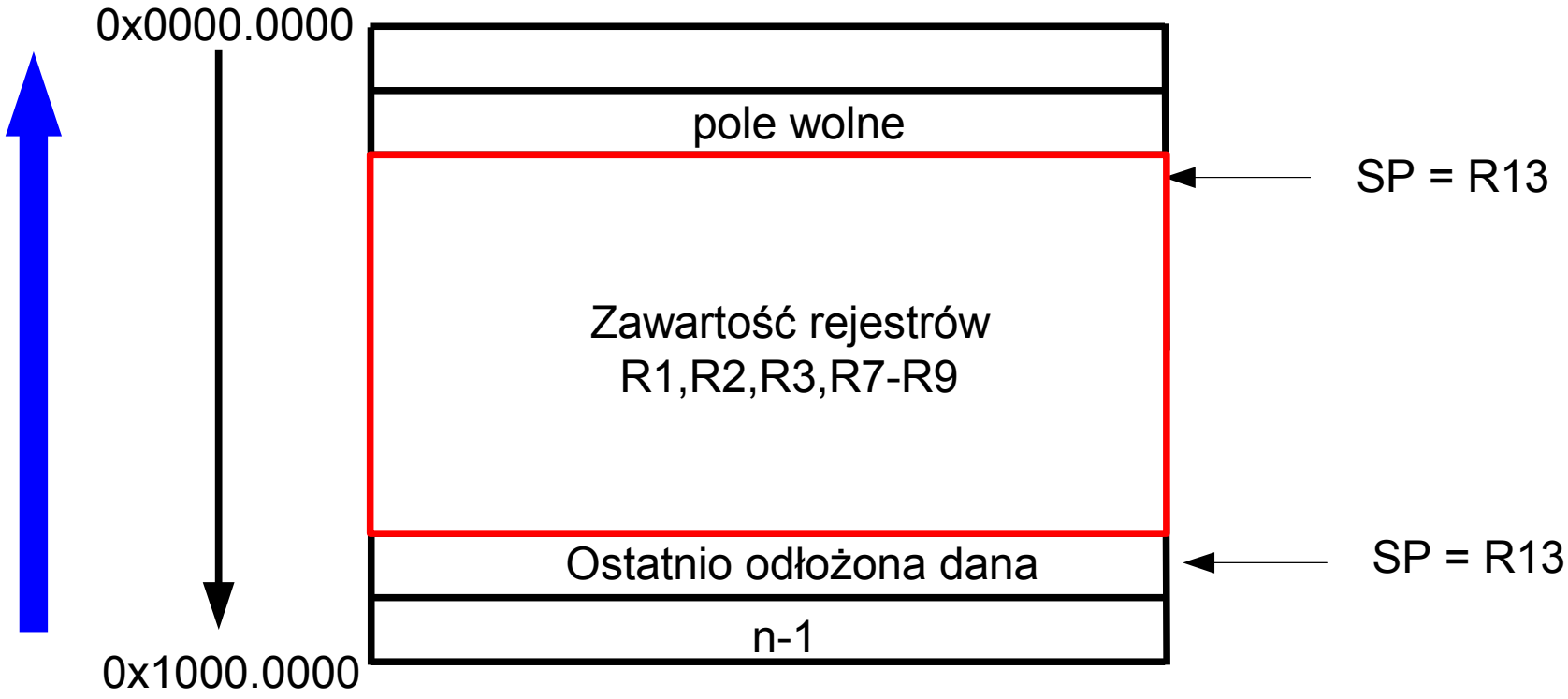
Stos (ang. stack lub LIFO Last-In, First-Out) - liniowa struktura danych, w której dane odkładane są na wierzch stosu i z wierzchołka stosu są zdejmowane. Ideę stosu danych można zilustrować jako stos położonych jedna na drugiej książek – nowy egzemplarz kładzie się na wierzch stosu i z wierzchu stosu zdejmuje się kolejne egzemplarze. Elementy stosu poniżej wierzchołka stosu można wyłącznie obejrzeć, aby je ściągnąć, trzeba najpierw po kolei ściągnąć to, co jest nad nimi

FIFO (ang. First In, First Out) - przeciwieństwem stosu LIFO jest kolejka, bufor typu FIFO (pierwszy na wejściu, pierwszy na wyjściu), w którym dane obsługiwane są w takiej kolejności, w jakiej zostały dostarczone (jak w kolejce do kasy)

First–in First–out (FIFO)

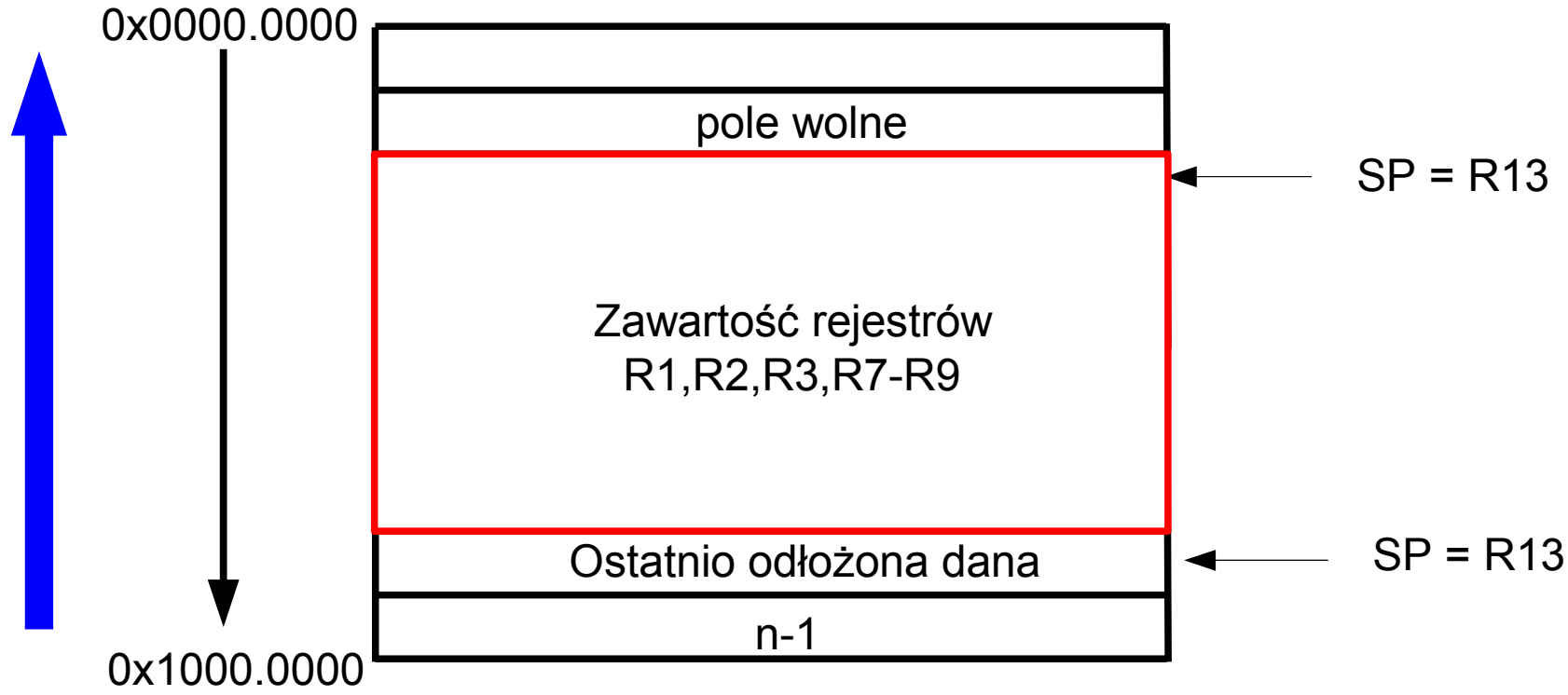


Struktura stosu – odłożenie danej na stos



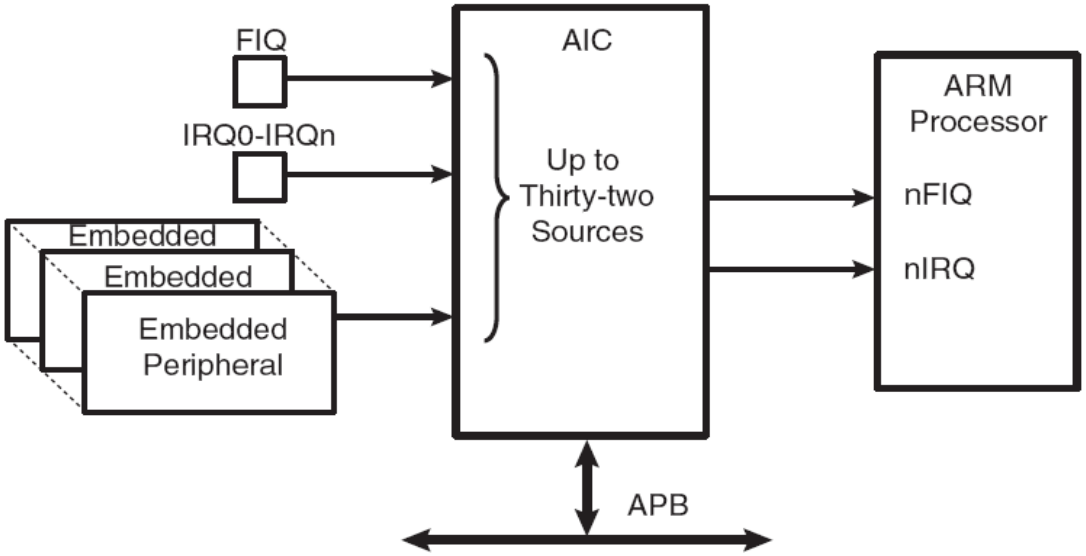

```
STMDB   SP!, {lista rejestrów}      | zmniejszenie SP o 24, odłożenie zawartości rejestrów  na
STMDB   SP!, {R1,R2,R3,R7-R9}         stos,
```

Struktura stosu – zdjęcie danej ze stosu




```
LDMIA   SP!, {lista rejestrów}      | zwiększenie SP o 24, odłożenie zawartości rejestrów  na
LDMIA   SP!, {R1,R2,R3,R7-R9}         stos,
```

Schemat blokowy sterownika przerwań procesora ARM



- Obsługa przerwań wektorowych,
- Obsługa do 32 przerwań zewnętrznych i wewnętrznych,
- Możliwość maskowania dowolnego przerwania,
- Obsługa przerwań nIRQ i szybkich nFIR (ang. Fast Interrupt Request),
- 8 poziomów priorytetów (0- najniższy, 7- najwyższy),
- Obsługa przerwań wyzwalanych poziomem lub zboczem.

Schemat blokowy sterownika przerwań procesora ARM

- Sterownik przerwań wykorzystuje zegar systemowy. Zegar doprowadzany jest przez cały czas pracy procesora (nie ma możliwości odcięcia zegara).
- Przerwania mogą zostać wykorzystane do wyprowadzenia procesora ze stanu uśpienia (Idle mode).
- Przerwanie o numerze 0 (FIQ) jest zawsze przerwaniem typu FIQ.
- Przerwanie o numerze 1 (SYS) sumą logiczną przerwań od urządzeń peryferyjnych procesora. W procedurze obsługi przerwania należy określić urządzenie/a zgłaszające przerwanie/a.
- Przerwania o numerach 2-31 (PID2-PID331) mogą zostać dołączone do urządzeń peryferyjnych (użytkownika) lub portów I/O.
- Sterownik obsługuje przerwania wyzwalane poziomem lub zboczem sygnału.

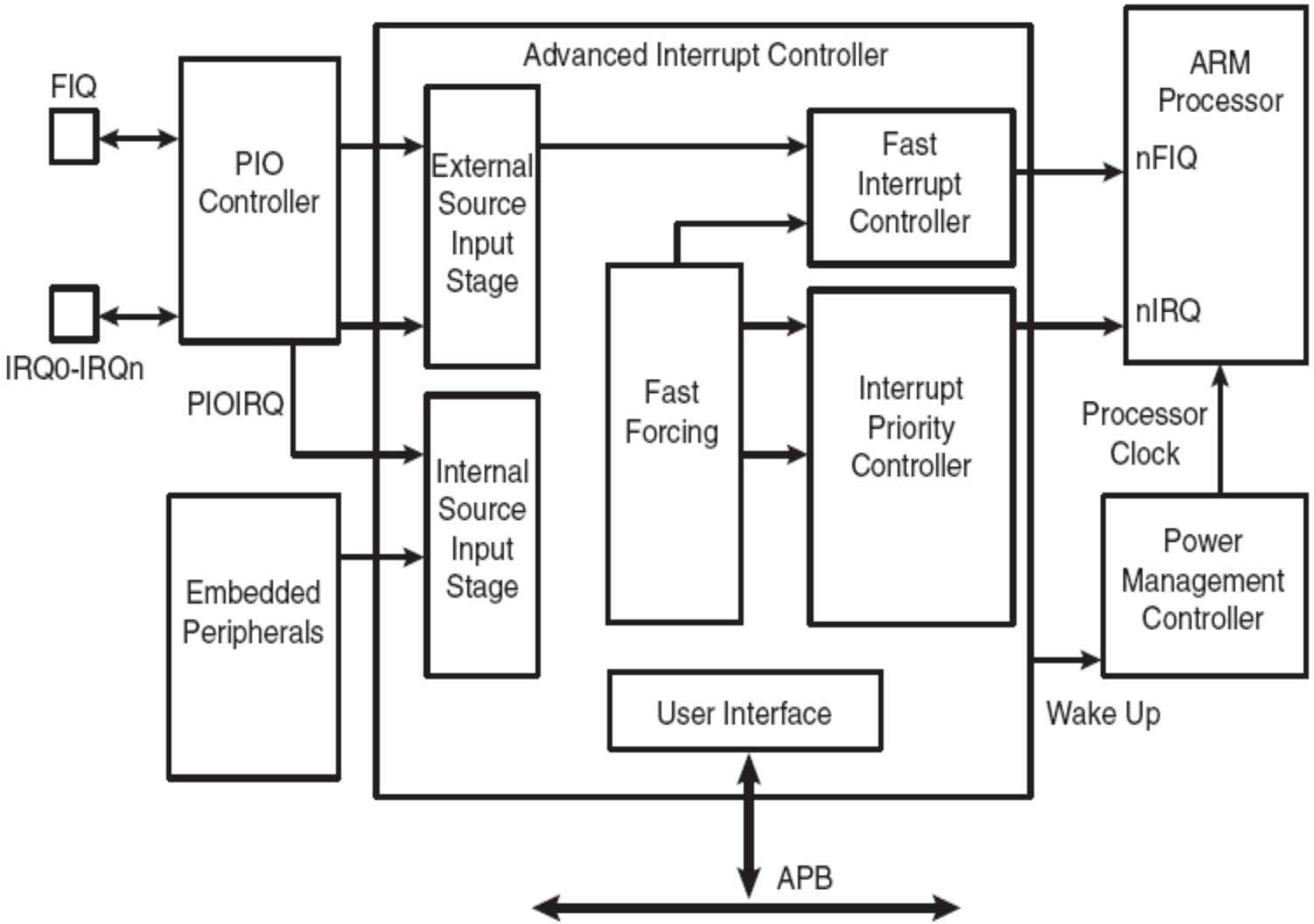
Przerwania współdzielone

Blok urządzeń systemowy (AT91C_ID_SYS) dysponuje jednym, wspólnym przerwaniem SYS (ang. shared interrupt) o numerze ID=1, które obejmuje następujące urządzenia:

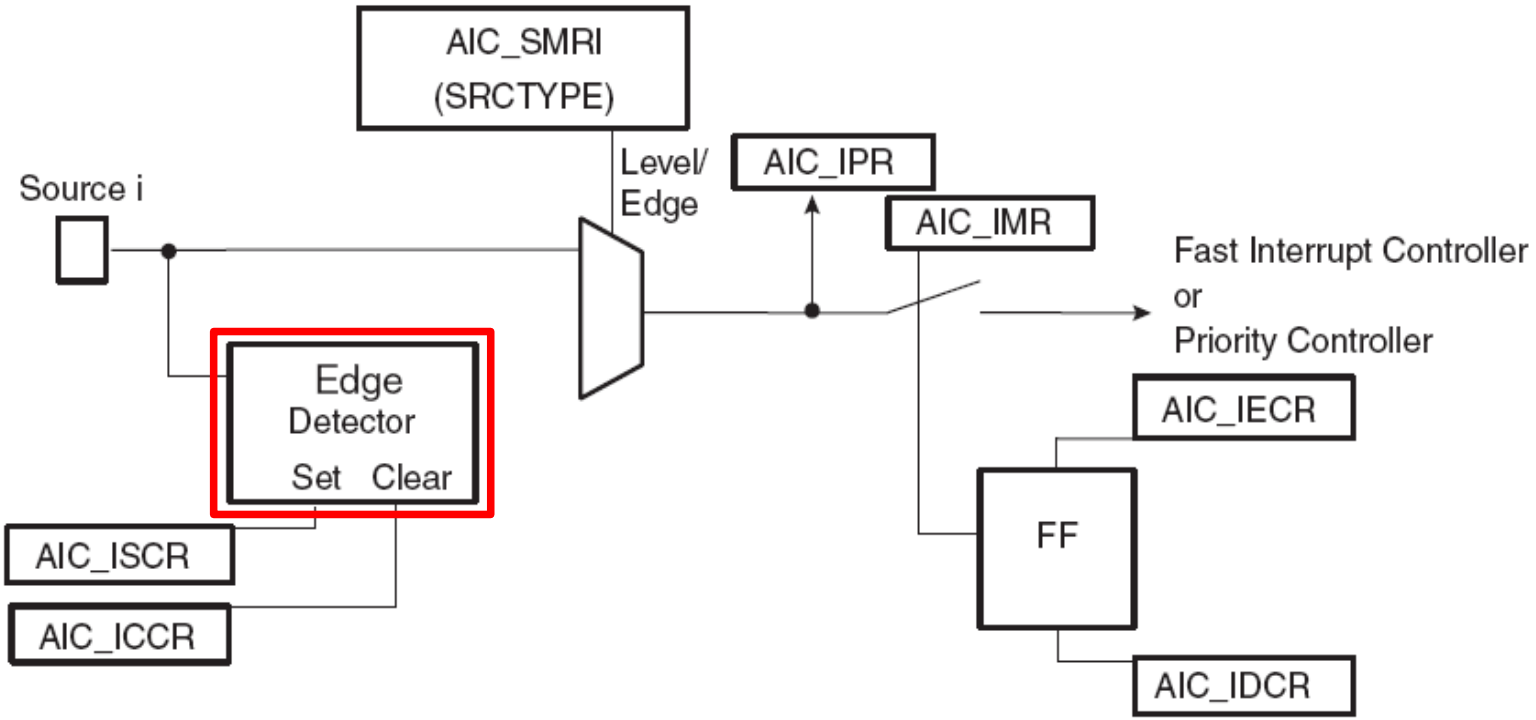
- timery PIT, RTT, WDT,
- interfejs diagnostyczny DBGU,
- Sterownik DMA PMC,
- Układ zerowania procesora RSTC,
- Sterownik pamięci MC.

W procedurze obsługi przerwania SYS należy sprawdzić kolejno stan wszystkich urządzeń, czy występują przerwania odmaskowane. Jeżeli przerwanie jest aktywne należy sprawdzić flagę sygnalizującą przerwanie w rejestrze statusu danego urządzenia. Jeżeli flaga jest ustawiona należy wykonać program związany z obsługą przerwania od danego urządzenia.

Szczególowy schemat blokowy sterownika AIC

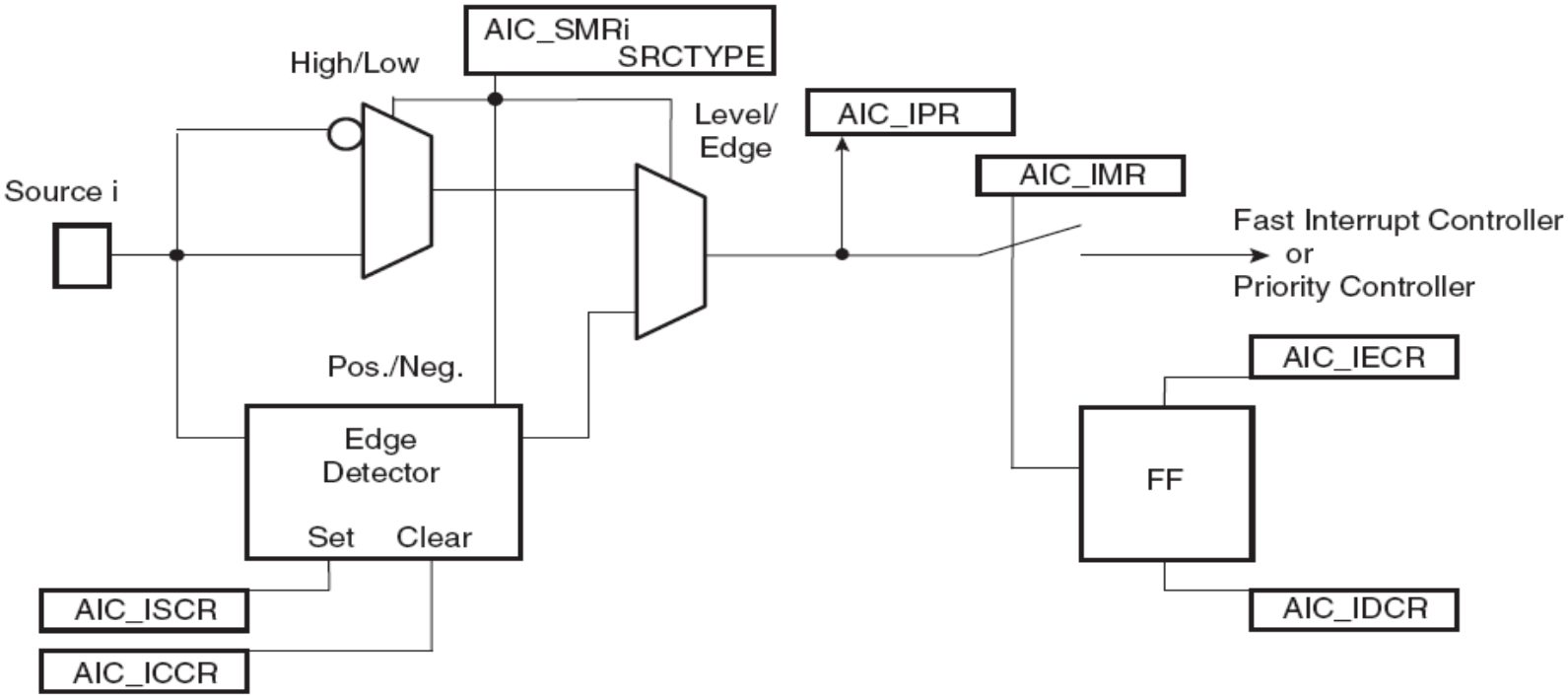


Przerwania wewnętrzne



- **Przerwania wewnętrzne wyzwalane tylko zboczem**
- Maska przerwań – AIC_IECR/IDCR (status → AIC_IMR),
- Wyczyszczenie flagi przerwania podczas odczytu rejestru AIC_IVR (przerwania FIQ → AIC_FVR),
- Status przerwań dostępny w rejestrze AIC_IPR (interrupt pending register)

Przerwania zewnętrzne



- Daje dodatkową możliwość wyboru zbocza opadającego/narastającego lub poziomu niskiego/wysokiego do generacji przerwań (rejestr SMR0-31).

Definicja numerów urządzeń peryferyjnych

```
// ********************************************************************************
//               PERIPHERAL ID DEFINITIONS FOR AT91SAM9263
// ********************************************************************************
#define AT91C_ID_FIQ    ( 0) // Advanced Interrupt Controller (FIQ)
#define AT91C_ID_SYS    ( 1) // System Controller
#define AT91C_ID_PIOA   ( 2) // Parallel IO Controller A
#define AT91C_ID_PIOB   ( 3) // Parallel IO Controller B
#define AT91C_ID_PIOCDE ( 4) // Parallel IO Controller C, Parallel IO Controller D, Parallel IO Controller E
#define AT91C_ID_US0    ( 7) // USART 0
#define AT91C_ID_US1    ( 8) // USART 1
#define AT91C_ID_US2    ( 9) // USART 2
#define AT91C_ID_MCI0   (10) // Multimedia Card Interface 0
#define AT91C_ID_MCI1   (11) // Multimedia Card Interface 1
#define AT91C_ID_CAN    (12) // CAN Controller
#define AT91C_ID_TWI    (13) // Two-Wire Interface
#define AT91C_ID_SPI0   (14) // Serial Peripheral Interface
```

ID=0, ID=30-31 przerwania zewnętrzne, pozostałe to przerwania wewnętrzne.

Rejestry sterownika przerwań (1)

Offset	Register	Name	Access	Reset
0x00	Source Mode Register 0	AIC_SMR0	Read-write	0x0
0x04	Source Mode Register 1	AIC_SMR1	Read-write	0x0
---	---	---	---	---
0x7C	Source Mode Register 31	AIC_SMR31	Read-write	0x0
0x80	Source Vector Register 0	AIC_SVR0	Read-write	0x0
0x84	Source Vector Register 1	AIC_SVR1	Read-write	0x0
---	---	---	---	---
0xFC	Source Vector Register 31	AIC_SVR31	Read-write	0x0
0x100	Interrupt Vector Register	AIC_IVR	Read-only	0x0
0x104	FIQ Interrupt Vector Register	AIC_FVR	Read-only	0x0
0x108	Interrupt Status Register	AIC_ISR	Read-only	0x0
0x10C	Interrupt Pending Register(3)	AIC_IPR	Read-only	0x0(1)
0x110	Interrupt Mask Register(3)	AIC_IMR	Read-only	0x0
0x114	Core Interrupt Status Register	AIC_CISR	Read-only	0x0
0x118 - 0x11C	Reserved	---	---	---
0x120	Interrupt Enable Command Register(3)	AIC_IECR	Write-only	---
0x124	Interrupt Disable Command Register(3)	AIC_IDCR	Write-only	---
0x128	Interrupt Clear Command Register(3)	AIC_ICCR	Write-only	---
0x12C	Interrupt Set Command Register(3)	AIC_ISCR	Write-only	---
0x130	End of Interrupt Command Register	AIC_EOICR	Write-only	---
0x134	Spurious Interrupt Vector Register	AIC_SPU	Read-write	0x0
0x138	Debug Control Register	AIC_DCR	Read-write	0x0
0x13C	Reserved	---	---	---
0x140	Fast Forcing Enable Register(3)	AIC_FFER	Write-only	---
0x144	Fast Forcing Disable Register(3)	AIC_FFDR	Write-only	---
0x148	Fast Forcing Status Register(3)	AIC_FFSR	Read-only	0x0
0x14C - 0x1E0	Reserved	---	---	---
0x1EC - 0x1FC	Reserved			

Rejestry sterownika przerwań

```
typedef struct _AT91S_AIC {
    AT91_REG    AIC_SMR[32];    // Source Mode Register
    AT91_REG    AIC_SVR[32];    // Source Vector Register
    AT91_REG    AIC_IVR;        // IRQ Vector Register
    AT91_REG    AIC_FVR;        // FIQ Vector Register
    AT91_REG    AIC_ISR;        // Interrupt Status Register
    AT91_REG    AIC_IPR;        // Interrupt Pending Register
    AT91_REG    AIC_IMR;        // Interrupt Mask Register
    AT91_REG    AIC_CISR;       // Core Interrupt Status Register
    ...
} AT91S_AIC, *AT91PS_AIC;
#define AT91C_BASE_AIC      (AT91_CAST(AT91PS_AIC)   0xFFFFF000) // (AIC)
    Base Address
```

Rejestry sterownika przerwań (2)

AIC_SMR[32]; // Source Mode Register – konfiguracja poziomu oraz sposobu wyzwalania

AIC_SVR[32]; // Source Vector Register – handlery obsługujące przerwania

AIC_IVR; // IRQ Vector Register – adres handlera na obsługiwanego przerwania

AIC_FVR; // FIQ Vector Register – adres handlera na obsługiwanego przerwania

AIC_ISR; // Interrupt Status Register – numer obsługiwanego przerwania

AIC_IPR; // Interrupt Pending Register – rejestr z flagami oczękujących przerwań 0-31

AIC_IMR; // Interrupt Mask Register – rejestr z maskami przerwań 0-31

AIC_CISR; // Core Interrupt Status Register – stan przerwań rdzenia IRQ/FIQ

AIC_IECR; // Interrupt Enable Command Register – rejestr uaktywniający przerwania

AIC_IDCR; // Interrupt Disable Command Register – rejestr wyłączający przerwania

AIC_ICCR; // Interrupt Clear Command Register – rejestr ustawiający flagi przerwań

AIC_ISCR; // Interrupt Set Command Register – rejestr kasujący flagi przerwań

AIC_EOICR; // End of Interrupt Command Register – koniec obsługi przerwania, ack.

AIC_SPU; // Spurious Vector Register – handler do przerwania fałszywego

Procedura obsługująca przerwanie od timera PIT i klawiatury

Ustawienie adresu funkcji (handlera) obsługującego przerwanie (adres 32-bitowy)

AT91C_BASE_AIC->AIC_SVR[AT91C_ID_SYS] = (unsigned long) TIMER_handler;

Procedura obsługi przerwania od timera

void TIMER_handler (void) {

jeżeli flaga włączająca przerwanie od timera INT_ENABLE jest ustawiona (rejestr PITC_PIMR) to odczyt rejestru PITC_PIVR - skasowanie flagi przerwania (odczyt rejestru statutowego PITC_PISR)

Potwierdzenie obsługi przerwania, rejestr AIC_EOICR

jeżeli nie to inne urządzenie peryferyjne zgłosiło przerwanie – odpowiednia reakcja

}

Procedura obsługi przerwania od klawiatury

void BUTTON_handler (void) {

jeżeli flaga na odpowiednim bicie rejestru PIO_ISR jest ustawiona to oznacza to wciśnięcie przycisku

- Odczyt rejestru statutowego PIO_ISR - skasowanie flagi przerwania
- Odczyt rejestru przedstawiającego stan portu (PIO_PDSR)
- Potwierdzenie obsługi przerwania, rejestr AIC_EOICR

}

Konfiguracja przerwań od klawiatury

Przyciski dołączone są do portu C – przerwania generowane przez układy wejściowe portu C/D/E (maska AT91C_ID_PIOCDE)

Konfiguracja przerwań od portów C/D/E:

1. Konfiguracja portów procesora jako porty wejściowe (przycisk lewy i prawy)
2. Wyłączenie przerwań generowanych przez porty C/D/E (rejestr AIC_IDCR, AT91C_ID_PIOCDE)
3. Ustawienie wskaźnika do procedury obsługującej przerwanie dla portów C/D/E w tablicy wektorów SVR (AIC_SVR[AT91C_ID_PIOCDE])
4. Konfiguracja poziomu i metody wyzwalania przerwania (rejestr AIC_SMR, wyzwalanie wysokim poziomem AT91C_AIC_SRCTYPE_EXT_HIGH_LEVEL oraz priorytet AT91C_AIC_PRIOR_HIGHEST)
5. Wyczyszczenie flagi przerwania portów C/D/E (rejestr AIC_ICCR)
6. Włącznie przerwań dla obu przycisków (rejestr PIO_IER)
7. Włączenie przerwania od portów C/D/E (AIC_IECR)
8. Włączenie portu IO

Konfiguracja przrwań od Timera PIT

Timer PIT generuje przerwania o numerze 1 – przerwania od urządzeń peryferyjnych (System Controller, maska AT91C_ID_SYS)

Konfiguracja przerwań od Timera PIT

1. Konfiguracja okresu timera, np. 5 ms
2. Wyłączenie przerwania od Timera PIT na czas konfiguracji (AIC_IDCR, przerwanie nr 1 - urządzenia peryferyjne procesora, stała AT91C_ID_SYS)
3. Ustawienie wskaźnika do procedury obsługującej przerwanie dla urządzeń peryferyjnych w tablicy wektorów AIC_SVR (AIC_SVR[AT91C_ID_SYS])
4. Konfiguracja poziomu i metody wyzwalania przerwania (rejestr AIC_SMR, wyzwalanie AT91C_AIC_SRCTYPE_INT_LEVEL_SENSITIVE, dowolny poziom, np. AT91C_AIC_PRIOR_LOWEST)
5. Wyczyszczenie flagi przerwania urządzeń peryferyjnych (rejestr AIC_ICCR)
6. Włącznie przerwania urządzeń peryferyjnych AT91C_ID_SYS (rejestr AIC_IECR)
7. Włączenie przerwania od Timera (AT91C_PITC_PITIEN)
8. Włączenie Timera PIT (AT91C_PITC_PITEN)
9. Wyzerowanie tzw. licznika lokalnego Timera (zmienna Local_Counter)

Funkcje obsługujące przerwania w języku C (1)

Deklaracja procedur obsługujących przerwania (kompilator GCC) wymaga użycia dyrektywy preprocesora __attribute__ ((interrupt("IRQ"))):

```
void INTButton_handler()__attribute__ ((interrupt("IRQ")));
void INTPIT_handler()__attribute__ ((interrupt("IRQ")));
void Soft_Interrupt_handler()__attribute__ ((interrupt("SWI")));
void Abort_Exception_handler()__attribute__ ((interrupt("ABORT")));
void Undef_Exception_handler()__attribute__ ((interrupt("UNDEF")));
```

Funkcja obsługująca przerwanie nie różni się od klaszycznej funkcji w języku C

```
void INTButton_handler() {
    // standard C function
}
```

Przerwania współdzielone

Blok urządzeń systemowy (AT91C_ID_SYS) dysponuje jednym, wspólnym przerwaniem SYS (ang. shared interrupt) o numerze ID=1, które obejmuje następujące urządzenia:

- timery PIT, RTT, WDT,
- interfejs diagnostyczny DBGU,
- Sterownik DMA PMC,
- Układ zerowania procesora RSTC,
- Sterownik pamięci MC.

W procedurze obsługi przerwania SYS należy sprawdzić kolejno stan wszystkich urządzeń, czy występują przerwania odmaskowane. Jeżeli przerwanie jest aktywne należy sprawdzić flagę sygnalizującą przerwanie w rejestrze statusu danego urządzenia. Jeżeli flaga jest ustawiona należy wykonać program związany z obsługą przerwania od danego urządzenia.

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

„Procesory ARM w systemach wbudowanych”

„Rodzina procesorów ARM”

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”