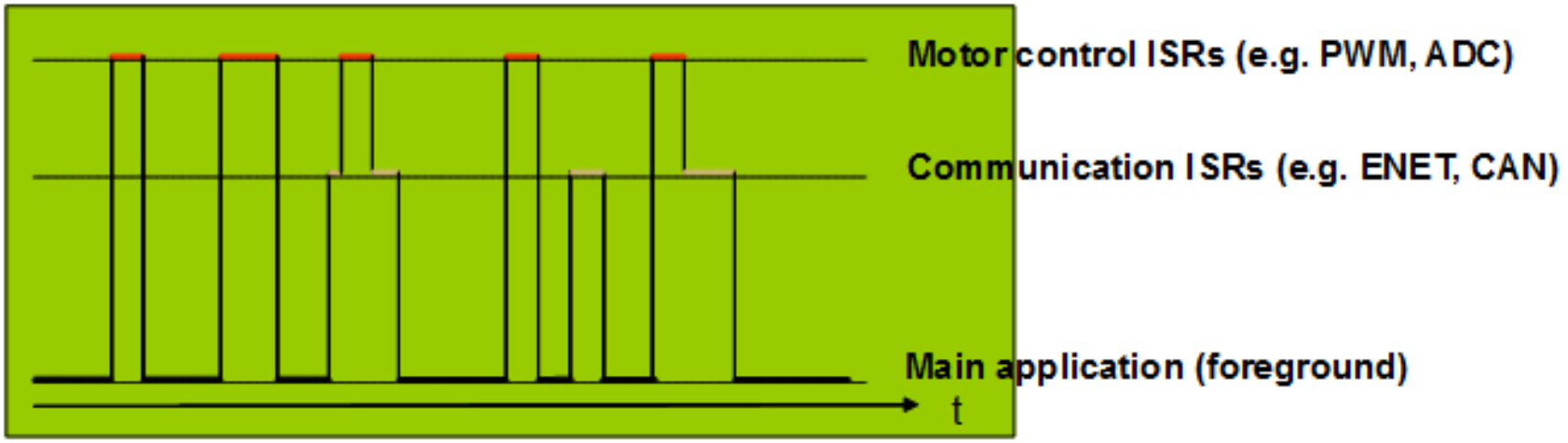
Wykład 9

Obsługa przerwań

Kontroler przerwań

- Kontroler NVIC udostępnia globalne maskowanie przerwań, ustawianie priorytetów i funkcji obsługi.
- Procesor LM3S6965 umożliwia obsługę 38 przerwań. Każde przerwanie może być indywidualnie maskowane, dodatkowo przerwanie rdzenia może być globalnie maskowane (bez wpływania na maski ustawione indywidualnie dla każdego z przerwań).



Kontroler przerwań

- Kontroler przerwań jest bezpośrednio sprzężony z rdzeniem mikrokontrolera.
- Kiedy procesor otrzymuje sygnał przerwania, NVIC udostępnia adres procedury obsługi przerwania bezpośrednio do procesora.
- Eliminuje to potrzebę odpytywania kontrolera NVIC, w celu określenia adresu obsługi i skoku do odpowiedniej funkcji. Dzięki temu redukowany jest czas obsługi przerwania.

Kontroler przerwań

Interrupt handling is micro-coded. No instruction overhead

Entry

Automatically pushes registers R0–R3, R12, LR, PSR, and PC onto the stack

In parallel, ISR is pre-fetched on the instruction bus.

ISR ready to start executing as soon as stack PUSH complete

A late-arriving interrupt will restart ISR pre-fetch, but state saving does not need to be repeated

Exit

Processor state is automatically restored from the stack

In parallel, interrupted instruction is pre-fetched ready for execution upon completion of stack POP

Stack POP can be interrupted, allowing new ISR to be immediately executed without the overhead of state saving

Kontroler przerwań

Interrupt handling is micro-coded. No instruction overhead

Entry

Automatically pushes registers R0–R3, R12, LR, PSR, and PC onto the stack

In parallel, ISR is pre-fetched on the instruction bus.

ISR ready to start executing as soon as stack PUSH complete

A late-arriving interrupt will restart ISR pre-fetch, but state saving does not need to be repeated

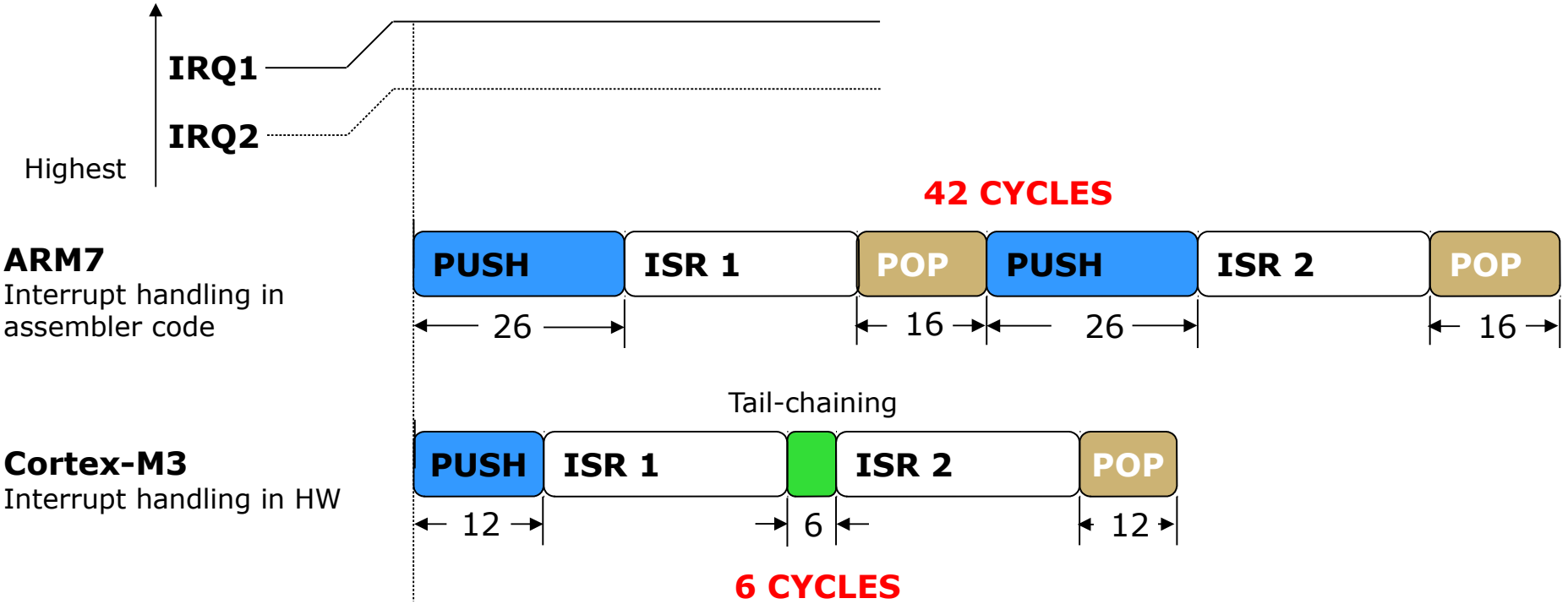
Exit

Processor state is automatically restored from the stack

In parallel, interrupted instruction is pre-fetched ready for execution upon completion of stack POP

Stack POP can be interrupted, allowing new ISR to be immediately executed without the overhead of state saving

Interrupt Latency - Tail Chaining



ARM7

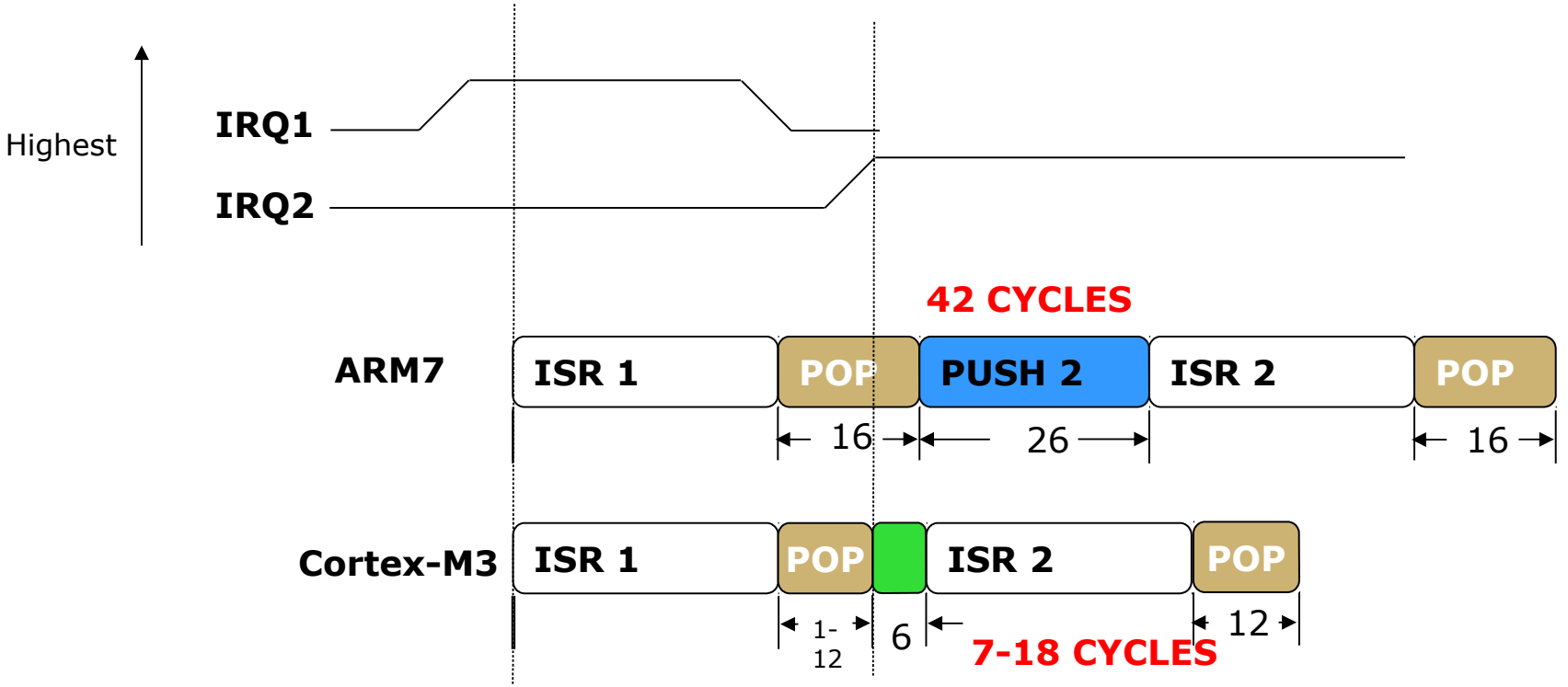
- 26 cycles from IRQ1 to ISR1 entered
  - Up to 42 cycles if LSM
  - 42 cycles from ISR1 exit to ISR2 entry
  - 16 cycles to return from ISR2

Cortex-M3

- 12 cycles from IRQ1 to ISR1 entered
  - 12 cycles if LSM
  - 6 cycles from ISR1 exit to ISR2 entry
  - 12 cycles to return from ISR2

Pre-emption…

Interrupt Latency – Pre-emption



ARM7

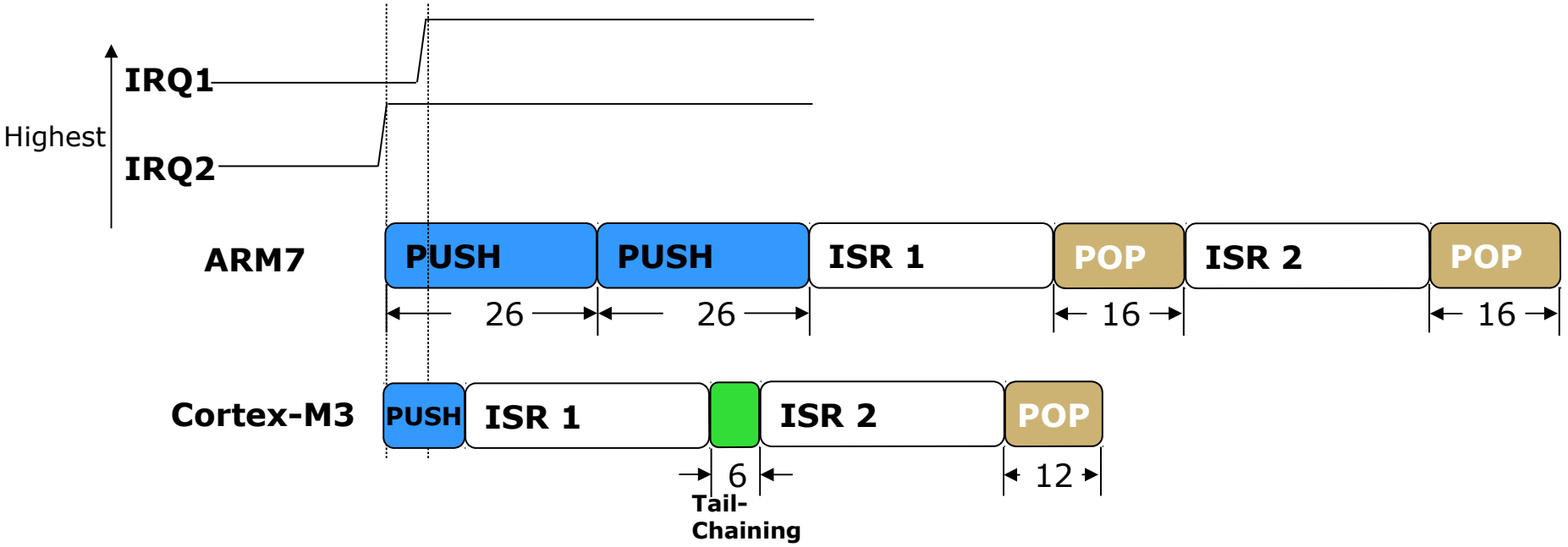
- Load Multiple uninterruptible, and hence the core must complete the POP and the full stack PUSH

Cortex-M3

- POP may be abandoned early if another interrupt arrives
- If POP is interrupted it only takes 6 cycles to enter ISR2 (Equivalent to Tail -chaining)

Late Arrival

Interrupt Latency – Late Arriving



ARM7

- 26 cycles to ISR2 entered
- Immediately pre-empted by IRQ1 and takes a further 26 cycles to enter ISR 1
- ISR 1 completes and then takes 16 cycles to return to ISR 2

Cortex-M3

- Stack push to ISR 2 is interrupted
- Stacking continues but new vector address is fetched in parallel
- 6 cycles from late-arrival to ISR1 entry
- Tail-chain into ISR 2

Peripherals…

Kontroler przerwań

- Priorytety kontrolera przerwań umożliwiają obsługę przerwań o wyższym priorytecie przed przerwaniami o niższym priorytecie, a także umożliwiają przerywanie przerwań o niższym priorytecie przez przerwania o wyższym priorytecie.
- Umożliwia to skrócenie czasu odpowiedzi (na przykład przerwanie systemowe czasie trwania 1ms nie jest blokowane przez trwające 1 s przerwanie o niższym priorytecie).
- Dodatkowo możliwe są podpriorytety dla przerwań o tym samym priorytecie.
- Dla takich przerwań nie ma możliwości wywłaszczenia, wykorzystywane jest do ich obsługi szeregowanie (tail chaining).

Tryby pracy procesora

- Procesor wspiera dwa tryby pracy, tryb Thread i tryb Handler:
  - tryb Thread jest uruchamiany po resecie i jest ustawiany po powrocie z obsługi wyjątku. W trybie tym może być uruchamiany kod uprzywilejowany (Privileged) lub kod użytkownika (User),
  - tryb Handler jest uruchamiany po wystąpieniu wyjątku. Kod uruchamiany w tym trybie jest kodem uprzywilejowanym.

Wyjątki, błędy i przerwania

- Wyjątki mogą być generowane poprzez wywołanie instrukcji generującej wyjątek, lub wyzwalane jako odpowiedź na zachowanie systemu takie jak:
  - przerwanie,
  - zarządzanie pamięcią,
  - wyrównywanie (alignment),
  - błąd magistrali,
  - wyjątek debugera.

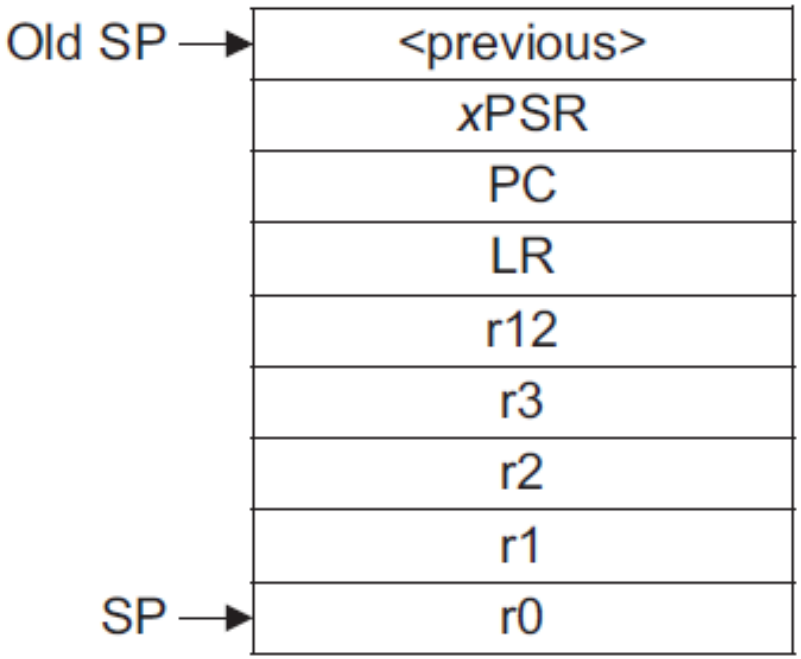
- Architektura umożliwia rozróżnienie wyjątków synchronicznych i asynchronicznych.

Obsługa wyjątków

- Procesor i kontroler przerwań ustalają pierwszeństwo i obsługują wszystkie wyjątki.
- Wyjątki obsługiwane są w trybie Handler.
- Stan procesora jest automatycznie składowany na stosie po wystąpieniu wyjątku i automatycznie odkładany po zakończeniu obsługi Interrupt Service Routine (ISR). Następujące rejestry są odkładane na stosie:
  - Program Counter (PC),
  - Processor Status Register (xPSR),
  - r0-r3,
  - r12,
  - Link Register (LR).

Obsługa wyjątków

- Rysunek przedstawia kolejność na stosie poszczególnych rejestrów.



- Wektor obsługi przerwania jest równolegle pobierany z zapisem stanu, co skraca czas obsługi wyjątku.

Typy wyjątków

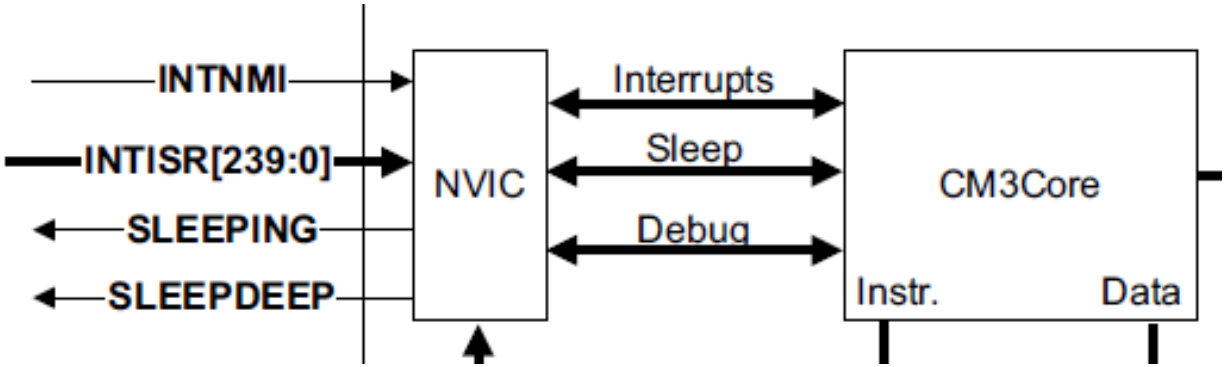
Typ wyjątku	Pozycja	Priorytet	Opis
-	0	-	Wierzchołek stosu jest ładowany w czasie resetu z pierwszej pozycji tablicy wektorów przerwań
Reset	1	-3(highest)	Wywoływany po włączeniu zasilania i miękkim resecie. Na pierwszej instrukcji zmniejszany na niższy priorytet (Thread mode). Asynchroniczny.
Non-maskable Interrupt	2	-2	Nie może być zatrzymany i wywłaszczony przez żaden wyjątek poza resetem. Asynchroniczny.
Hard Fault	3	-1	Wszystkie rodzaje błędów. Asynchroniczny.
Memory Management	4	ustawiany	Problemy z Memory Protection Unit (MPU), włączając błędy dostępu, niedostępność. Asynchroniczny.
Bus Fault	5	ustawiany	Błędy w czasie pobierania instrukcji, błędy dostępu do pamięci.
Usage Fault	6	ustawiany	Błędy użycia, takie jak niezdefiniowana instrukcja, próba nieprawidłowej zmiany stanu. Synchroniczny.

Typy wyjątków

Typ wyjątku	Pozycja	Priorytet	Opis
-	7-10	-	Zarezerwowane.
SVCall	11	ustawiany	Wywoływanie funkcji systemowej instrukcją SVC. Synchroniczny.
Debug monitor	12	ustawiany	Monitor debugera, kiedy nie jest wstrzymany. Synchroniczny.
-	13	-	Zarezerwowany.
PendSV	14	ustawiany	Żądanie obsługi systemowej. Asynchroniczny. Wyzwalany tylko programowo.
SysTick	15	ustawiany	Zegar systemowy jest uruchomiony. Asynchroniczny.
External interrupt	16 i większy	ustawiany	Wywoływany z zewnątrz rdzenia, INTISR[239:0] i przekazywany przez NVIC. Asynchroniczny.

Kontroler przerwań

- Kontroler przerwań (Nested Vectored Interrupt Controller (NVIC)) umożliwia obsługę do 240 przerwań z programowym przyporządkowywaniem priorytetów przerwań.



Kontroler przerwań

- Można przyporządkować poziomy od 0 do 255 do przerwań, poprzez zapisanie ośmiu bitów pól PRI-N w rejestrze Interrupt Priority Register. Sprzętowy priorytet zmniejsza się wraz ze zwiększaniem numeru przerwania.
- Priorytet 0 jest najwyższym priorytetem, a 255 najniższym. Ustawiane priorytety nadpisują sprzętowe priorytety. Jeżeli na przykład przyporządkujemy priorytet 1 do IRQ[0] i prorytet 0 do IRQ[31], to wtedy IRQ[31] ma wyższy priorytet niż IRQ[0].
- Najwyższy priorytet możliwy do ustawienia przez użytkownika wynosi 0. Jest on przetwarzany jako czwarty w kolejności po Reset, NMI i Hard Fault. Priorytet 0 jest domyślnym priorytetem dla wszystkich ustawialnych priorytetów.

Kontroler przerwań

- Ustawianie priorytetów programowo nie wpływa na priorytet Reset, NMI i Hard Fault. Mają one zawsze wyższy priorytet od przerwań zewnętrznych.
- Jeżeli przyporządkujemy ten sam poziom priorytetu dla dwóch lub więcej przerwań, to wtedy ich sprzętowy priorytet (niższy numer pozycji) będzie decydował o kolejności przyjmowania przerwań przez procesor.
- Na przykład jeżeli przerwania portów IRQ[0] (GPIO Port A) i IRQ[1] (GPIO Port B) mają ustawiony priorytet 1, wtedy IRQ[0] ma wyższy priorytet niż IRQ[1].

Przerwania wyzwalane poziomem i zboczem

- Procesor umozliwia generowanie przerwań wyzwalanych poziomem i zboczem sygnału.
- Przerwanie wyzwalane poziomem jest utrzymywane do momentu, kiedy zostanie skasowane w rejestrach ISR.
- W celu wykrycia zbocza przerwania jest ono próbkowane przy narastającym zboczu zegara HCLK.
- Dla przerwań wyzwalanych poziomem, jeżeli sygnał zgłoszenia nie jest zdjęty do zakończenia obsługi przerwania, to przerwanie jest ponownie aktywowane.
- Obsługa tego typu jest przydatna przy buforach FIFO i urządzeniach z buforowaniem danych, ponieważ jeden sygnał zgłoszenia przerwania może być wykorzystany do wielokrotnego wywołania funkcji obsługi przerwania, aż do opróżnienia bufora.

Przerwania wyzwalane poziomem i zboczem

- Przerwanie wyzwalane zboczem może być odblokowane w trakcie obsługi, dzięki czemu może być w stanie aktywnym i obsługi w tym samym czasie.
- Drugie przerwanie nie jest obsługiwane w trakcie obsługi pierwszego. Jeżeli zgłoszenie nastąpi w ostatnim cyklu to jest ono zapamiętywane.

Tabela wektorów przerwań w pliku startup.s.

```
;*******************************************************************************
;
;
; The vector table.
;
;*******************************************************************************
;
        EXPORT  __Vectors
__Vectors
        DCD     StackMem + Stack           ; Top of Stack
        DCD     Reset_Handler              ; Reset Handler
        DCD     NmiSR                    ; NMI Handler
        DCD     FaultISR                 ; Hard Fault Handler
        DCD     IntDefaultHandler          ; MPU Fault Handler
        DCD     IntDefaultHandler          ; Bus Fault Handler
        DCD     IntDefaultHandler          ; Usage Fault Handler
        DCD     0                       ; Reserved
        DCD     0                       ; Reserved
        DCD     0                       ; Reserved
        DCD     0                       ; Reserved
        DCD     IntDefaultHandler          ; SVCall Handler
        DCD     IntDefaultHandler          ; Debug Monitor Handler
        DCD     0                       ; Reserved
        DCD     IntDefaultHandler          ; PendSV Handler
        DCD     IntDefaultHandler          ; SysTick Handler
        DCD     IntDefaultHandler          ; GPIO Port A
        DCD     IntDefaultHandler          ; GPIO Port B
        DCD     IntDefaultHandler          ; GPIO Port C
        DCD     IntDefaultHandler          ; GPIO Port D
        DCD     IntDefaultHandler          ; GPIO Port E
```

Tabela wektorów przerwań w pliku startup.s.

```
DCD     IntDefaultHandler       ; UART0
DCD     IntDefaultHandler       ; UART1
DCD     IntDefaultHandler       ; SSI
DCD     IntDefaultHandler       ; I2C
DCD     IntDefaultHandler       ; PWM Fault
DCD     IntDefaultHandler       ; PWM Generator 0
DCD     IntDefaultHandler       ; PWM Generator 1
DCD     IntDefaultHandler       ; PWM Generator 2
DCD     IntDefaultHandler       ; Quadrature Encoder
DCD     IntDefaultHandler       ; ADC Sequence 0
DCD     IntDefaultHandler       ; ADC Sequence 1
DCD     IntDefaultHandler       ; ADC Sequence 2
DCD     IntDefaultHandler       ; ADC Sequence 3
DCD     IntDefaultHandler       ; Watchdog
DCD     IntDefaultHandler       ; Timer 0A
DCD     IntDefaultHandler       ; Timer 0B
DCD     IntDefaultHandler       ; Timer 1A
DCD     IntDefaultHandler       ; Timer 1B
DCD     IntDefaultHandler       ; Timer 2A
DCD     IntDefaultHandler       ; Timer 2B
DCD     IntDefaultHandler       ; Comp 0
DCD     IntDefaultHandler       ; Comp 1
DCD     IntDefaultHandler       ; Comp 2
DCD     IntDefaultHandler       ; System Control
DCD     IntDefaultHandler       ; Flash Control
```

Tabela wektorów przerwań w pliku startup.s.

```
DCD    IntDefaultHandler        ; GPIO Port F
DCD    IntDefaultHandler        ; GPIO Port G
DCD    IntDefaultHandler        ; GPIO Port H
DCD    IntDefaultHandler        ; UART2 Rx and Tx
DCD    IntDefaultHandler        ; SSI1 Rx and Tx
DCD    IntDefaultHandler        ; Timer 3 subtimer A
DCD    IntDefaultHandler        ; Timer 3 subtimer B
DCD    IntDefaultHandler        ; I2C1 Master and Slave
DCD    IntDefaultHandler        ; Quadrature Encoder 1
DCD    IntDefaultHandler        ; CAN0
DCD    IntDefaultHandler        ; CAN1
DCD    0                    ; Reserved
DCD    IntDefaultHandler        ; Ethernet
DCD    IntDefaultHandler        ; Hibernate
```

Rejestry kontrolera przerwań

- Rejestry kontrolera NVIC zawarte są w przestrzeni rejestrów systemowych (System Control space).
- Przestrzeń kontrolera przerwań można podzielic następujaco:
  - 0xE000E000 - 0xE000E00F. Interrupt Type Register
  - 0xE000E010 - 0xE000E0FF. System Timer
  - 0xE000E100 - 0xE000ECFF. NVIC
  - 0xE000ED00 - 0xE000ED8F. System Control Block, including:
  - — CPUID
  - — System control, configuration, and status
  - — Fault reporting
  - 0xE000EF00 - 0xE000EF0F. Software Trigger Exception Register
  - 0xE000EFD0 - 0xE000EFFF. ID space.

Tabela rejestrów

Nazwa rejestru	Typ	Adres	Wartość po resecie
Interrupt Control Type Register	Read-only	0xE000E004	a
SysTick Control and Status Register	Read/write	0xE000E010	0x00000000
SysTick Reload Value Register	Read/write	0xE000E014	Unpredictable
SysTick Current Value Register	Read/write clear	0xE000E018	Unpredictable
SysTick Calibration Value Register	Read-only	0xE000E01C	STCALIB
Irq 0 to 31 Set Enable Register	Read/write	0xE000E100	0x00000000
. . . . .			
. . . . .			
Irq 224 to 239 Set Enable Register	Read/write	0xE000E11C	0x00000000
Irq 0 to 31 Clear Enable Register	Read/write	0xE000E180	0x00000000
Irq 224 to 239 Clear Enable Register	Read/write	0xE000E19C	0x00000000
Irq 0 to 31 Set Pending Register	Read/write	0xE000E200	0x00000000
. . . . .			
. . . . .			
Irq 224 to 239 Set Pending Register	Read/write	0xE000E21C	0x00000000
Irq 0 to 31 Clear Pending Register	Read/write	0xE000E280	0x00000000
. . . . .			
. . . . .			
Irq 224 to 239 Clear Pending Register	Read/write	0xE000E29C	0x00000000
Irq 0 to 31 Active Bit Register	Read-only	0xE000E300	0x00000000
. . . . .			
. . . . .			
Irq 224 to 239 Active Bit Register	Read-only	0xE000E31C	0x00000000

Tabela rejestrów

Nazwa rejestru	Typ	Adres	Wartość po resecie
Irq 0 to 31 Priority Register	Read/write	0xE000E400	0x00000000
. . . . .			
. . . . .			
Irq 236 to 239 Priority Register	Read/write	0xE000E4F0	0x00000000
CPUID Base Register	Read-only	0xE000ED00	0x411FC231
Interrupt Control State Register	Read/write or read-only	0xE000ED04	0x00000000
Vector Table Offset Register	Read/write	0xE000ED08	0x00000000
Application Interrupt/Reset Control Register	Read/write	0xE000ED0C	0x00000000b
System Control Register	Read/write	0xE000ED10	0x00000000
Configuration Control Register	Read/write	0xE000ED14	0x00000000
System Handlers 4-7 Priority Register	Read/write	0xE000ED18	0x00000000
System Handlers 8-11 Priority Register	Read/write	0xE000ED1C	0x00000000
System Handlers 12-15 Priority Register	Read/write	0xE000ED20	0x00000000
System Handler Control and State Register	Read/write	0xE000ED24	0x00000000
Configurable Fault Status Registers	Read/write	0xE000ED28	0x00000000
Hard Fault Status Register	Read/write	0xE000ED2C	0x00000000
Debug Fault Status Register	Read/write	0xE000ED30	0x00000000
Mem Manage Address Register	Read/write	0xE000ED34	Unpredictable
Bus Fault Address Register	Read/write	0xE000ED38	Unpredictable
Auxiliary Fault Status Register	Read/write	0xE000ED3C	0x00000000

Interrupt Controller Type Register

- Odczyt rejestru Interrupt Controller Type Register pozwala określić liczbę przerwań obsługiwaną przez kontroler NVIC.

- Adres rejestru, rodzaj dostępu i stan po resecie:
  - Adres 0xE000E004
  - Dostęp Read-only
  - Stan po resecie Zależy od zdefiniowanej liczby przerwań.

Interrupt Controller Type Register

Table 8-2 Interrupt Controller Type Register bit assignments

Bits	Field	Function
[31:5]	-	Reserved.
[4:0]	INTLINESNUM	Total number of interrupt lines in groups of 32: b00000 = 0...32[a] b00001 = 33...64 b00010 = 65...96 b00011 = 97...128 b00100 = 129...160 b00101 = 161...192 b00110 = 193...224 b00111 = 225...256a

a. The processor only supports between 1 and 240 external interrupts.

Interrupt Set-Enable Registers

- Rejestry Interrupt Set-Enable Registers są wykorzystywane do:
  - odblokowania przerwań,
  - określenia które przerwania są odblokowane.
- Każdy bit odpowiada jednemu z przerwań. Ustawienie bitu odpowiadającego przerwaniu powoduje jego uaktywnienie.
- Kasowanie bitów rejestru Interrupt Set-Enable Register bit nie wpływa na aktualnie aktywne przerwania. Zapobiega jedynie następnym aktywacją.
- Adres rejestru, rodzaj dostępu i stan po resecie:
  - Adres 0xE000E100-0xE000E11C
  - Dostęp Read/write
  - Stan po resecie 0x00000000

Interrupt Set-Enable Registers

- Bit Pole Funkcja
- [31:0] SETENA Bitu odblokowujące przerwanie.
- Dla operacji zapisu:
  - 1 = odblokowanie przerwania
  - 0 = bez wpływu.
- Dla operacji odczytu:
  - 1 = przerwanie odblokowane
  - 0 = przerwanie zablokowane
-
- Wpisanie 0 do SETENA nie daje żadnego rezultatu. Odczyt bitów zwraca aktualny stan. Reset zeruje bity SETENA.

Interrupt Clear-Enable Registers

- Rejestry Interrupt Clear-Enable Registers są wykorzystywane do:
  - zablokowania przerwań,
  - określenia które przerwania są zablokowane.
- Każdy bit odpowiada jednemu z przerwań. Ustawienie bitu odpowiadającego przerwaniu powoduje jego zablokowanie.
- Adres rejestru, rodzaj dostępu i stan po resecie:
  - Adres 0xE000E180-0xE000E19C
  - Dostęp Read/write
  - Stan po resecie 0x00000000

Interrupt Clear-Enable Registers

- Bits Field Function
- [31:0] CLRENA Interrupt set enable bits.
- Dla operacji zapisu:
  - 1 = zablokowanie przerwania
  - 0 = bez wpływu.
- Dla operacji odczytu:
  - 1 = przerwanie odblokowane
  - 0 = przerwanie zablokowane
- Wpisanie 0 do CLRENA nie daje żadnego rezultatu. Odczyt bitów zwraca aktualny stan.

Interrupt Set-Pending Register

- Rejestry Interrupt Set-Pending Register są wykorzystywane do:
  - wymuszania odłożenie przerwania
  - okreslania, które przerwania są oczekujace.
- Każdy bit rejestru odpowiada jednemu z przerwań. Ustawienie bitu Interrupt
- Set-Pending Register odkłada odpowiadające danemu bitowi przerwanie.
- Zapis do rejestru Interrupt Set-Pending Register nie ma wpływu na oczekujące i zablokowane przerwania.
- Adres rejestru, rodzaj dostępu i stan po resecie::
  - Adres 0xE000E200-0xE000E21C
  - Dostęp Read/write
  - Stan po resecie 0x00000000

Interrupt Clear-Pending Register

- Rejestry Interrupt Clear-Pending Register są wykorzystywane do:
  - kasowania odłożenie przerwania
  - określania, które przerwania są oczekujące.
- Każdy bit rejestru odpowiada jednemu z przerwań. Ustawienie bitu Interrupt
- Clear-Pending Register ustawia odpowiadające danemu bitowi przerwanie w stan nieaktywny.
- Zapis do rejestru Interrupt Clear-Pending Register nie ma wpływu na aktywne przerwania, chyba że są aktualnie odłożone.
- Adres rejestru, rodzaj dostępu i stan po resecie:
  - Adres 0xE000E280-0xE000E29C
  - Dostęp Read/write
  - Stan po resecie 0x00000000

Active Bit Register

- Odczyttywanie rejestru Active Bit Register pozwala określić, które przerwania są aktywne. Każda z flag odpowiada jednemu przerwaniu.
- Adres rejestru, rodzaj dostępu i stan po resecie::
  - Adres 0xE000E300-0xE00031C
  - Dostęp Read-only
  - Stan po resecie 0x00000000

- Bit Pole Funkcja
- [31:0] ACTIVE Flagi:
  - 1 = przerwania aktywne lub wywłaszczone i odłożone,
  - 0 = przerwanie nieaktywne lub odłożone.

Interrupt Controller API

- Zestaw funkcji obsługi przerwań pozwala z poziomu języka C na obsługę kontrolera przerwań NVIC.
- Udostępnione funkcje pozwalają na blokowywanie i odblokowywanie przerwań, ustawianie funkcji obsługi przerwań i ustawianie priorytetów przerwań.
- Funkcje obsługi przerwań mogą być ustawiane na dwa sposoby:
  - statycznie w czasie kompilacji,
  - dynamicznie w czasie pracy programu.
- Jeżeli przerwania są statycznie konfigurowane, to muszą być odblokowane przez funkcję IntEnable() zanim procesor otrzyma przerwanie.
- Alternatywnie przerwanie może być skonfigurowane w czasie pracy przez funkcję IntRegister(), przerwanie także musi być odblokowane, jednak wykonuje to funkcja rejestrująca, a nie aplikacja użytkownika.

Interrupt Controller API

- Konfiguracja w czasie pracy wymaga, żeby tablica adresów funkcji obsługi przerwań była umieszczona w 1kB pamięci SRAM (zazwyczaj na początku).
- W celu umieszczenia w pamięci konieczne jest wykorzystanie skryptu linkera.
- Nie jest to możliwe w wersji szkolnej środowiska. programowania.
- Funkcje drajwera obsługi przerwań zawarte są w pliku src/interrupt.c wraz z plikiem nagłówkowym src/interrupt.h zawierającym definicje funkcji API do wykorzystania w aplikacji.

Funkcje API obsługi przerwań

- void IntDisable (unsigned long ulInterrupt)
- void IntEnable (unsigned long ulInterrupt)
- void IntMasterDisable (void)
- void IntMasterEnable (void)
- long IntPriorityGet (unsigned long ulInterrupt)
- unsigned long IntPriorityGroupingGet (void)
- void IntPriorityGroupingSet (unsigned long ulBits)
- void IntPrioritySet (unsigned long ulInterrupt, unsigned char ucPriority)
- void IntRegister (unsigned long ulInterrupt, void (#pfnHandler) (void))
- void IntUnregister (unsigned long ulInterrupt)

Przykład wykorzystania funkcji API

```
//
// Funkcja obsługi przerwania.
//
extern void IntHandler(void);
//
// Rejestracja funkcji dla przerwania 5.
//
IntRegister(5, IntHandler);
//
// Odblokowanie przerwania 5.
//
IntEnable(5);
//
// Odblokowanie przerwań.
//
IntMasterEnable();
```

Program przykładowy

```
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
int main(void)
{
unsigned long ulPeriod;
SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_OSC_MAIN |
SYSCTL_XTAL_8MHZ);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0);
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
TimerConfigure(TIMER0_BASE, TIMER_CFG_32_BIT_PER);
ulPeriod = (SysCtlClockGet() / 10) / 2;
TimerLoadSet(TIMER0_BASE, TIMER_A, ulPeriod -1);
IntMasterEnable();
IntEnable(INT_TIMER0A);
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
TimerEnable(TIMER0_BASE, TIMER_A);
while(1)
{
}
}
```

Program przykładowy

```
void Timer0IntHandler(void)
{
// Clear the timer interrupt
TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
// Read the current state of the GPIO pin and
// write back the opposite state
if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_0))
{
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
}
else
{
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 1);
}
}
```

Program przykładowy

```
IntDefaultHandler,                      // ADC Sequence 2
IntDefaultHandler,                      // ADC Sequence 3
IntDefaultHandler,                      // Watchdog timer
Timer0IntHandler,                       // Timer 0 subtimer A
IntDefaultHandler,                      // Timer 0 subtimer B
IntDefaultHandler,                      // Timer 1 subtimer A
```

```
37 // External declaration for the reset handler that is to be called when the
38 // processor is started
39 //
40 //*****************************************************************************
41 extern void _c_int00(void);
42 extern void Timer0IntHandler(void);
43
44 //*****************************************************************************
```