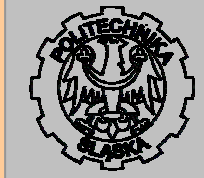


Studia Podyplomowe

EFEKTYWNE UŻYTKOWANIE ENERGII ELEKTRYCZNEJ

w ramach projektu

Śląsko-Małopolskie Centrum Kompetencji Zarządzania Energią

Urządzenia i oprogramowanie automatyki przemysłowej w sterowaniu i monitorowaniu systemów energetycznych

dr inż. Krzysztof Kołek

Systemy sterowania zużyciem energii w przemyśle i budownictwie

Urządzenia i oprogramowanie automatyki przemysłowej w sterowaniu i monitorowaniu systemów energetycznych

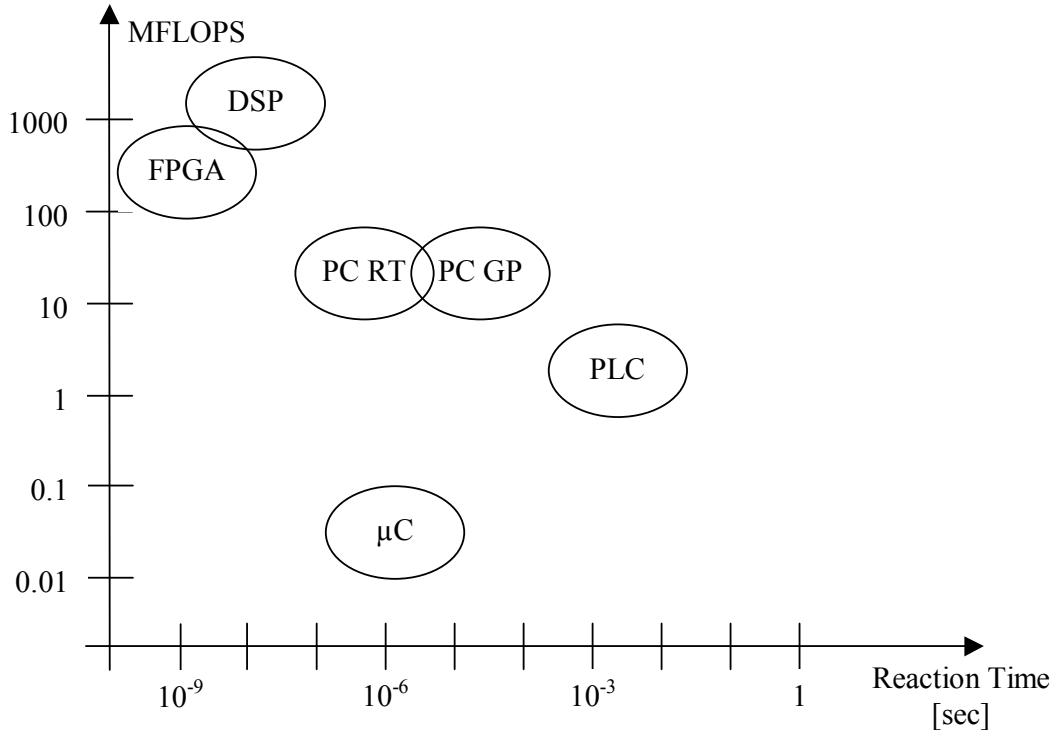
dr inż. Krzysztof Kołek

Katedra Automatyki i Inżynierii Biomedycznej AGH

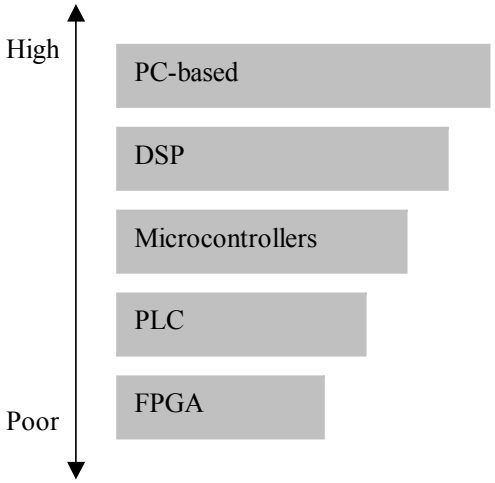
Plan

Ogólna charakterystyka
Przemysłowe komputery PC
Połączenia szeregowe RS-232, RS-422 i RS-485
Magistrala CAN
System czasu rzeczywistego
Systemy operacyjne czasu rzeczywistego
SCADA / HMI
PLC, pętla prądowa, SoftPLC
Sieci przemysłowe, modyfikacje Ethernet
Zastosowanie FPFA w systemach pomiarowo-sterujących

Platformy sprzętowe



Elastyczność programowa



Czasy reakcji

<1 µs zabezpieczenia układów elektronicznych

5 µs pozycjonowanie podczas drukowania metodą offsetową (0,1 mm przy prędkości 20 m/s)

55 µs synchronizacja czujników w podstacjach energetycznych (1º dla 50Hz)

100 µs sterowanie aktywnym zawieszeniem samochodu

100 µs sterowania w robotyce

1.6 ms algorytmy zabezpieczeń w podstacjach energetycznych

10 ms sterowanie układami mechatronicznymi, np. wahadło na wózku

20 ms czas zamykania/otwierania wyłączników wysokoprądowych

200 ms akceptowalny czas odpowiedzi na komendę operatora

1s akceptowalny czas odświeżania ekranu operatorskiego

3 s akceptowalny czas pojawiania się nowego ekranu operatorskiego

1 min akceptowalny czas wznowienia pracy po poważnej usterce

>5min dynamika dużych procesów cieplnych

PC

Przemysłowe wersje komputerów PC

Obudowy o odpowiednim IPxx (odporność na pył i wilgoć, najlepiej IP68 – chroni przed pyłem i może pracować zanurzone w wodzie)

Odporność na wstrząsy, dyski SSD

Podniesione wymagania dotyczące temperatury pracy. Chłodzenie z dodatkowymi filtrami lub bez wiatraków mechanicznych (szczelność)

Gniazda z mocowaniami o dużej niezawodności

Zasilanie typu hot-swap

Często przeznaczone do montażu w szafach lub na listwach DIN

Mechanizm *watchdog timer*

Cena kilkukrotnie wyższa od komputera biurowego

PC – magistrala PC104

PC104 – magistrala umożliwiająca rozszerzanie komputera o dodatkowe karty I/O. Magistrala posiada 120 sygnałów

Nadzorowana przez PC/104 Consortium

Nie posiada płyty głównej. Nowe moduły stosuje się, łącząc z poprzednimi

Rozmiar modułów: 3.55 × 3.775 cala

Podstawowe moduły to zasilacz oraz moduł procesora z peryferiami

Dodatkowe moduły to: czytniki kart zewnętrznej pamięci, moduły komunikacyjna (RS232, RS485, Ethernet, CAN, we/wy cyfrowe z optoizolacją, przetworniki A/C i C/A

PCM-3355 [1]

- Energooszczędny procesor AMD LX800 500 MHz lub LX600 366 MHz
- Do 1 GB DDR RAM
- Wbudowana podstawka pod kartę CompactF
- Praca bezwentylatorowa
- Karta sieciowa Ethernet 10/100
- 24-bit TFT LCD
- 2 x RS-232, 1 x RS-422/485
- 2 x USB 2.0
- 1 x SATA (max. 66 MB/s)

RS-232

Zdefiniowany w 1962r i ciągle używany w zastosowaniach przemysłowych. Pierwotnie przeznaczony do komunikacji za pomocą modemu telefonicznego, stracił to znaczenie i wykorzystywany jest do bezpośredniego łączenia dwóch komputerów

Na każdej linii wyłącznie jeden nadajnik i odbiornik

Zaprojektowany do komunikacji komputer / drukarka, terminal – krótkie przewody

Praca w trybie full-duplex

Szybkość do około 1Mbit/s

Sygnały typu *single-ended (unbalanced)* – wartość logiczna sygnału zależy od potencjału linii względem masy

Brak zgodności z TTL/LVTTL/CMOS – wymaga konwerterów. Duże bipolarne zmiany sygnału polepszają odporność na zakłócenia ale utrudniają zasilanie

RS-232

TxD, RxD, SG – sygnały nadawania, odbioru oraz sygnałowej masy; logika negatywna

RTS (Request To Send) żądanie nadawania danych

CTS (Clear to Send) – gotowość modemu – potwierdzenie przyjęcia RTS

DTR (Data Terminal Ready) – gotowość do pracy terminala

DSR (Data Set Ready) – gotowość do pracy modemu

DCD (Data Carier Detected) – modem połączył się z innym modemem

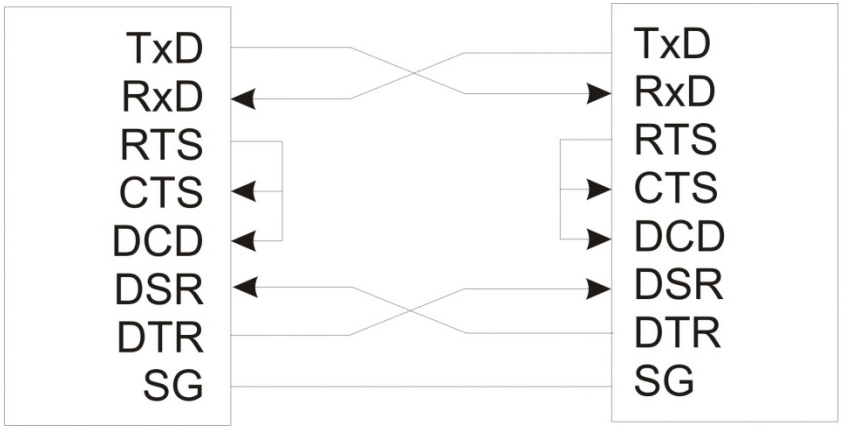
Sygnały sterujące mają logikę pozytywną

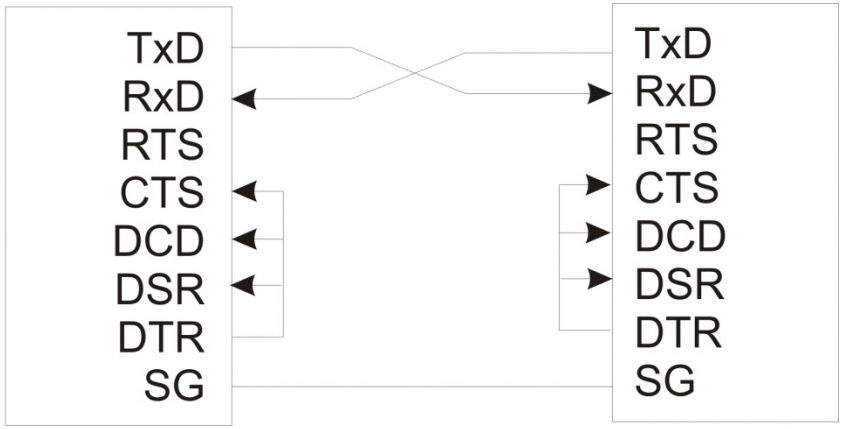
Sekwencja sygnałów sterujących:

- DTR i DSR w stanie 1 – gotowość do pracy terminala i modemu
- Terminal ustawia RTS na ’1’
- Modem (inne urządzenie) odpowiada ustawiając CTS na ’1’
- Wysyłane są dane

RS-232

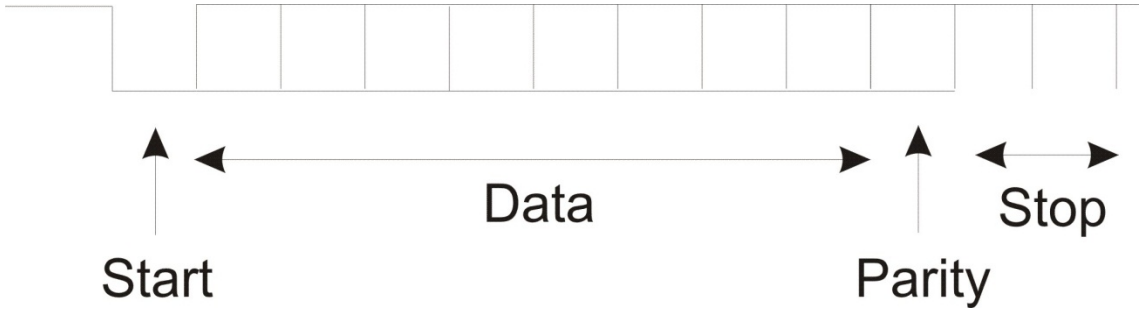
Sposoby połączenia komputer/komputer (PLC)





RS-232

Transmisja asynchroniczna. Popularne prędkości: 300,600,1200, 2400, 4800, 9600, 19 200, 38 400, 57 600, 115 200, 128 000 b/s. Dużo wolniej niż np. USB



Bit Start, 4-9 bitów znaku, opcjonalnie parzystość/nieparzystość, 1, 1.5 lub 2 bity Stop

Protokół XON – XOFF: 2 znaki sterują przepływem XON (11H) oraz XOFF (13H). Odbiornik znakiem XON sygnalizuje gotowość do odbioru; znak XOFF wstrzymuje transmisję

RS-422

Na każdej linii jeden nadajnik i do 10 odbiorników (multi-drop, ale nie multi-point ponieważ odbiorniki nie mogą współdzielić linii w celu nadawania)

Każdy sygnał to 2 przewody

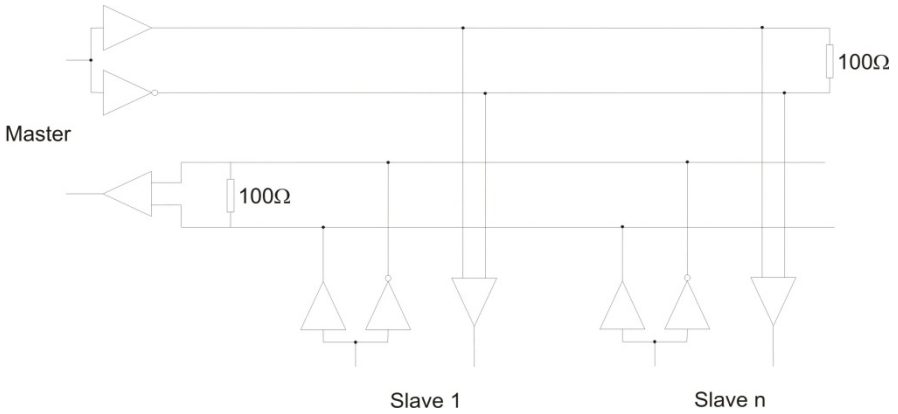
Maksymalna szybkość do około 10Mbit/s

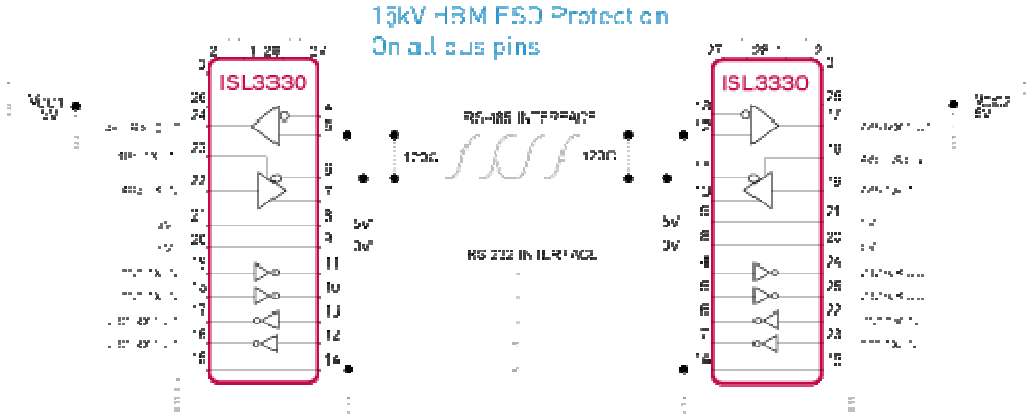
Sygnały typu *differencial (balanced)* – wartość logiczna sygnału zależy od różnicy potencjałów na dwóch przewodach

Sygnały *differential* są mniej podatne na zakłócenia oraz dryft masy, mają większą odległość transmisji

Konverter RS-232 / RS-422, 485 :

RS-422





RS-485

Do 32 urządzeń może komunikować się po tej samej linii

Posiada wyjścia trójstanowe – komunikacja po wspólnej linii

Komunikacja typu multi-point

Rozwiązywanie *data collisions* – np. nadajnik *master* rozpoczyna nadawanie od adresu. Odbiorniki nasłuchują, a odbiornik o nadawanym adresie „w locie” przełącza się na nadawanie. Każdy *slave* posiada unikalny adres i odpowiada wyłącznie na żądania urządzenia *master*

Oporniki terminujące dopasowują do impedancji falowej linii

RS-485

Platformy sprzętowe

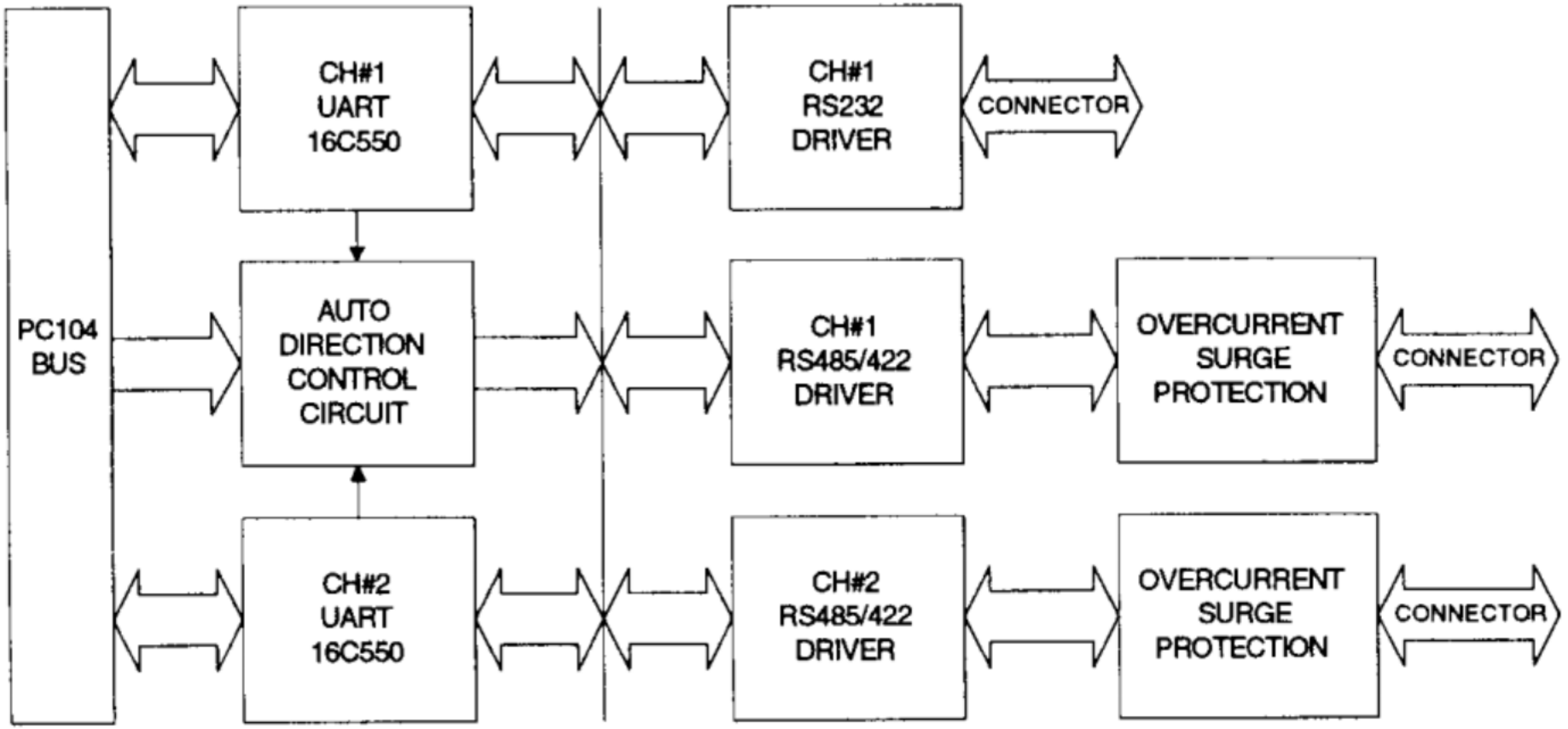
	RS-232	**RS-422**	**RS485**
Rodzaj sygnału	Single-ended	Differencial Multi-drop	Differential Multi-point
Nadajniki/obdiorniki	1 nadajnik/1 odbiornik	1 nadajnik/10 odbiorników	31 nadajniki/32 odbiorniki
Max dł. Przewodu	50 ft	4000ft	4000ft
Max. szybkość	1 Mbit/s	10 Mbit/s	10 Mbit/s
Max Vout	+/-25V	-0.25V / +6 V	-7V / +12 V
Poziom sygnału	+/-5V / +/-15V	+/-2V	+/-1.5V
Rez. obciążenia	3k – 7k	100	54
Vint odbiornika	+/-15V	+/-10V	-7V / +12V
Rez. odbiornika	3k – 7k	>= 4k	>=12k

PCM-3610 [2]

- Dwa porty z podwójnym złączem 10 pinowym.
- Port 1: RS-232/422/485.
- Port 2: RS-422/485.
- 16C550 UART z 16B FIFO.
- Szybkość od 50 do 56,000 bps.
- Dystans do 4000 ft/1.2 km.

PCM-3610 [2]

Optical isolation



CAN Controller Area Network

Magistrala szeregowa pierwotnie stworzona dla samochodów – powstanie wynikło ze wzrostu sterowników podzespołów samochodu i konieczności koordynacji ich pracy. Często stosowana w innych gałęziach przemysłu

Prędkość do 1Mbit/s

Protokół zorientowany na przesyłanie wiadomości (nie strumieni binarnych). Wiadomości są krótkie – maksimum 8 bajtów

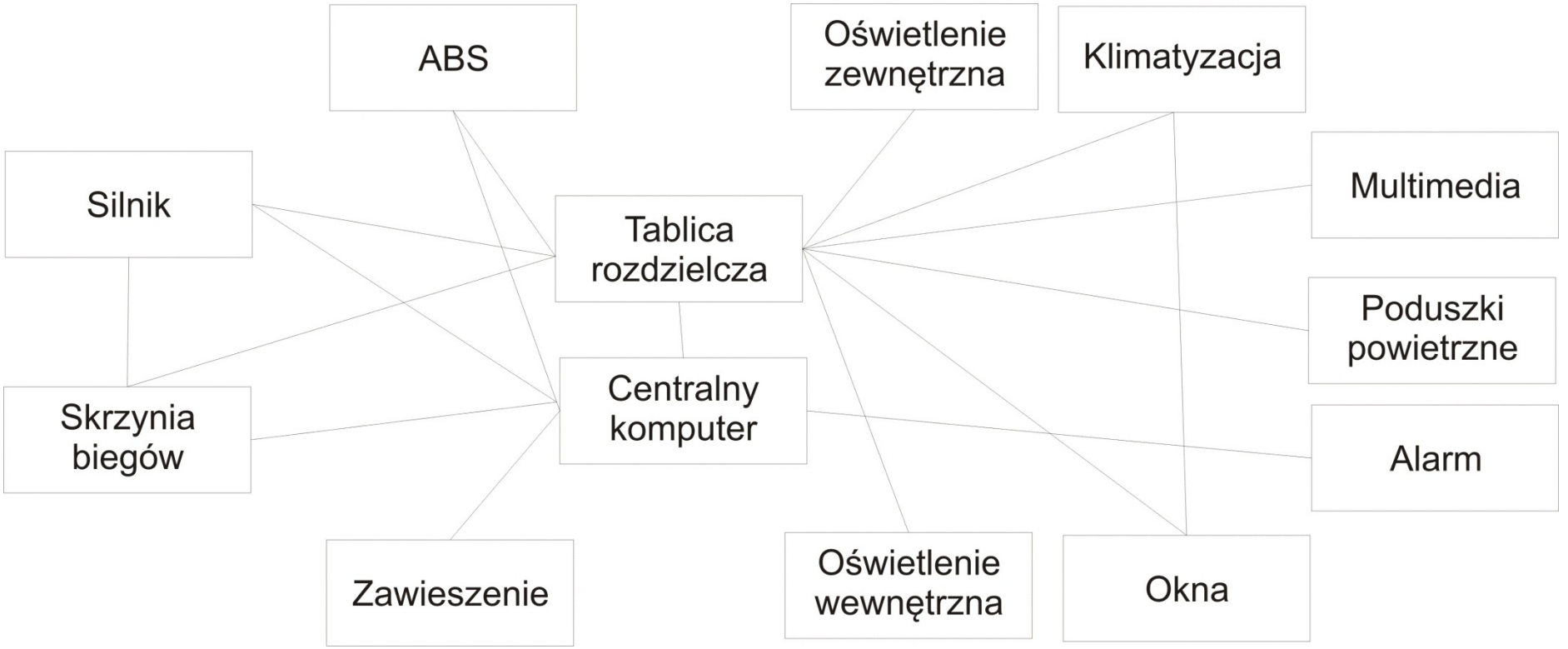
CAN pracuje w trybie broadcast – wszystkie węzły odbierają wszystkie komunikaty, reagując tylko na komunikaty dla nich przeznaczone

Węzły podłączone są do magistrali na zasadzie *wired-and*. Jeżeli węzeł wystawi logiczne 0, to magistrala przechodzi w stan 0

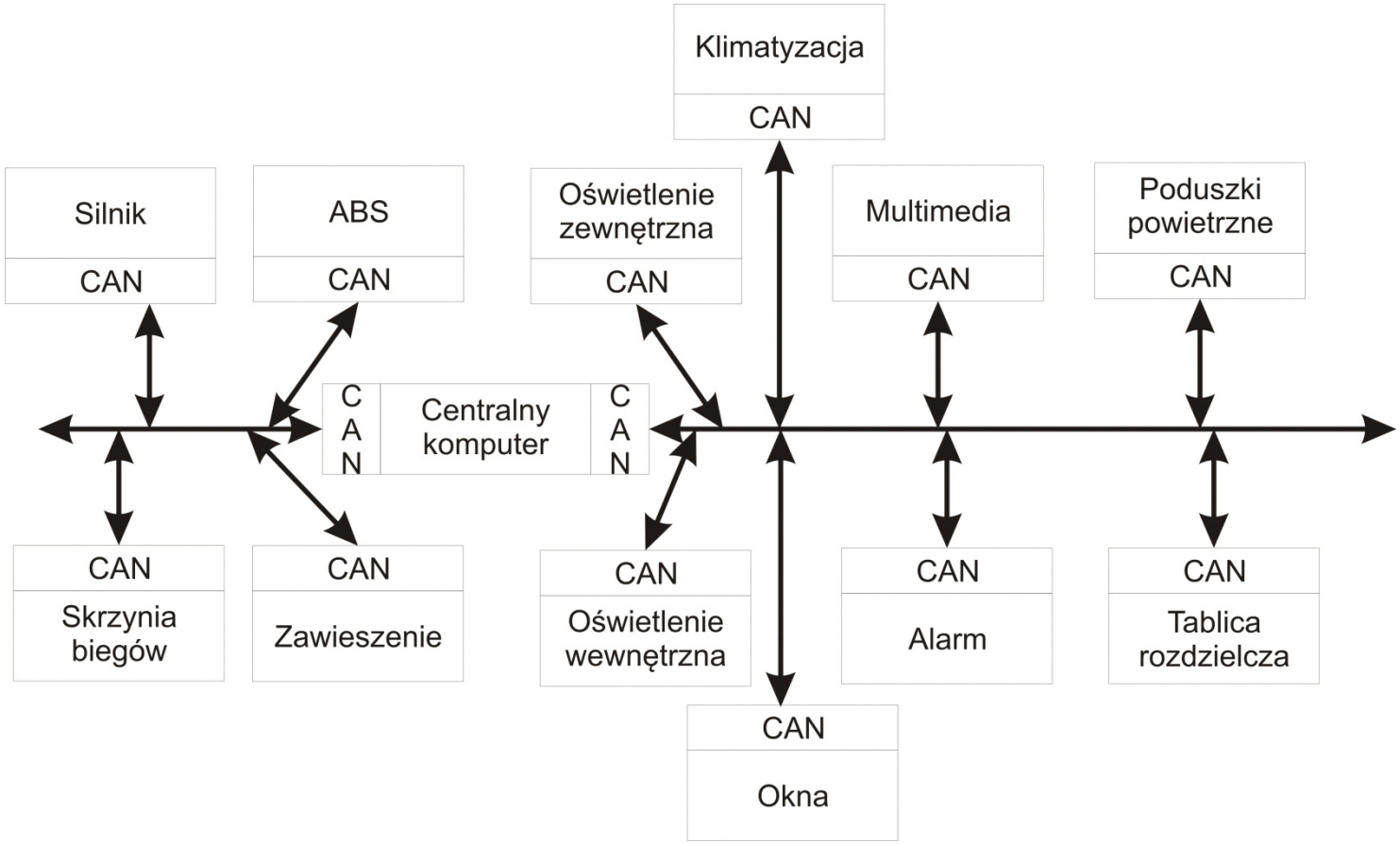
Transmisja różnicowa parą przewodów CAN_L i CAN_H. Terminowane opornikiem 120Ω na obu końcach magistrali

Kodowanie NRZ (Non Return to Zero) – 1 to stan wysoki napięcia, 0 – stan niski. Brak zmian może powodować rozsynchronizowanie.

Platformy sprzętowe



CAN


Klimatyzacja
CAN
Silnik
CAN
ABS
CAN
Oświetlenie zewnętrzna
CAN
Multimedia
CAN
Poduszki powietrzne
CAN
C A N
Centralny komputer
C A N
CAN
Skrzynia biegów
CAN
Zawieszenie
CAN
Oświetlenie wewnętrzna
CAN
Alarm
CAN
Tablica rozdzielcza
CAN
Okna

CAN

SOF	Identifier	RTR	IDE	r0	DLC	Data	CRC	ACK	EOF	IFS

SOF – Start of Frame

Identifier – priorytet wiadomości / typ danych. Długość 11 bitów CAN 2.0A lub 29 bitów CAB 2.0B (w tym 2 bity kontrolne)

RTR – Remote Transmission Request

IDE – typ identyfikatora 11 (CAN 2.0A)/29 (CAB 2.0B) bitów

DLC – Data Length Code

Data – do 8 bajtów danych

CRC – Cyclic Redundant Check

ACK – potwierdzenie ustawiane przez odbiorcę (**lub odbiorców**) ramki

EOF – End of Frame

IFS – Intermission Frame Space; minimalna liczba ‚1' separujących komunikaty

CAN

4 rodzaje ramek: *data*, *remote*, *error* i *overload*. Powyżej przedstawiona ramka danych

Ramka *remote*: podobna do ramki danych. Zamarkowana bitem RTR i nie posiada danych. Przeznaczeniem zamówienie transmisji z określonego węzła

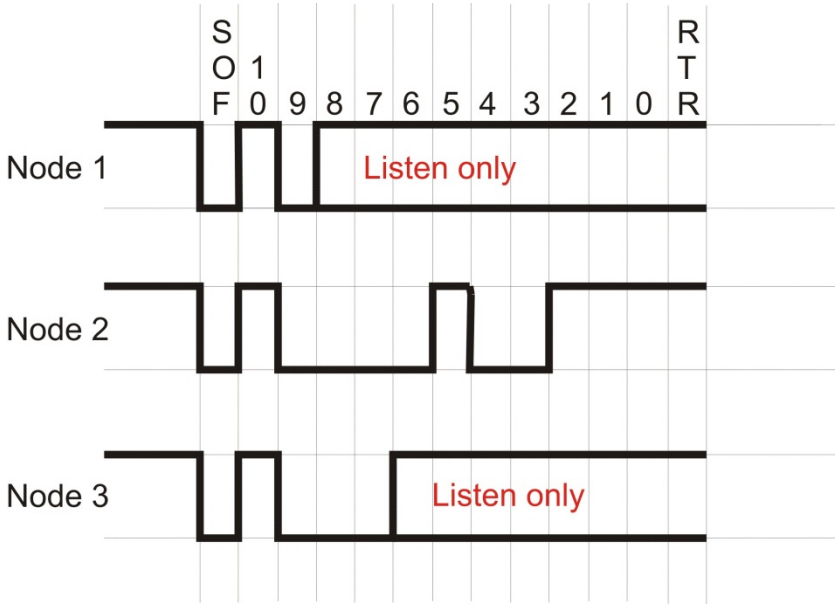
Ramka *error*: sygnalizuje wystąpienie błędu, po którym nastąpi próba retransmisji komunikatu

Ramka *overload*: sygnalizuje przeciążenie węzłą. Sugeruje spowolnienie komunikatów

3 klasy cieci:

- Class A – do 10 kbps
- Class B – 10 kbps do 125bkps
- Class C – 125 kbps – 1 Mbps

CAN



W przypadku konfliktu wygrywa identyfikator o najniższym numerze

Węzły otrzymują identyfikatory podczas konfiguracji

Nie ma jawnego adresu. Występuje wyłącznie identyfikator

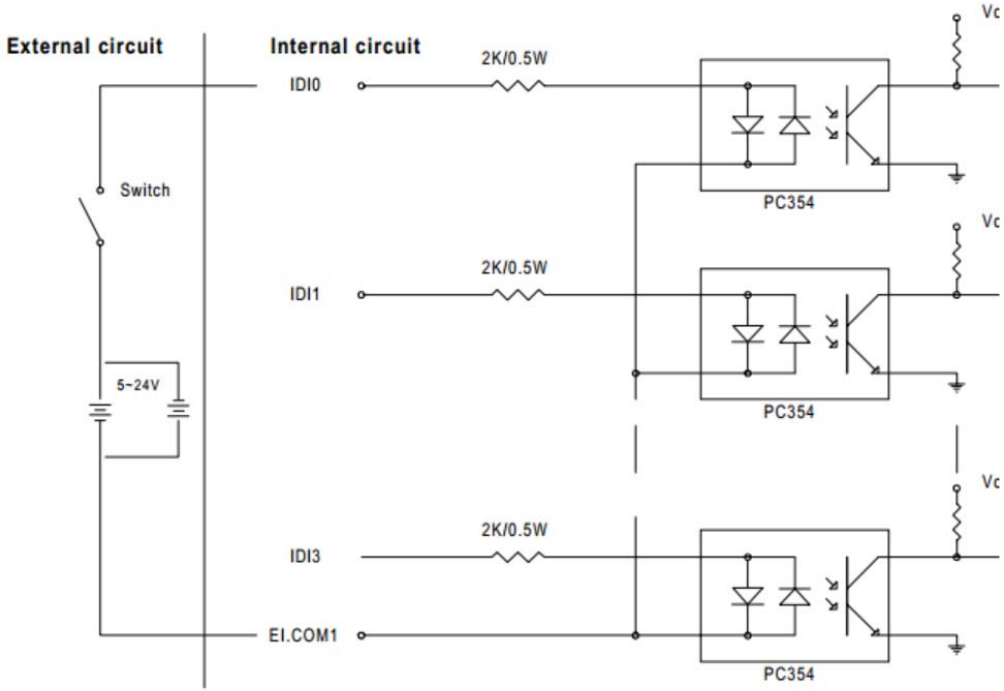
Konflikt rozgrywane poprzez funkcję *wired-and*. Węzły przegrywając natychmiast przechodzą w stan nasłuchiwania, a nadaje WYŁĄCZNIE węzeł o najwyższym priorytecie (najniższy identyfikator)

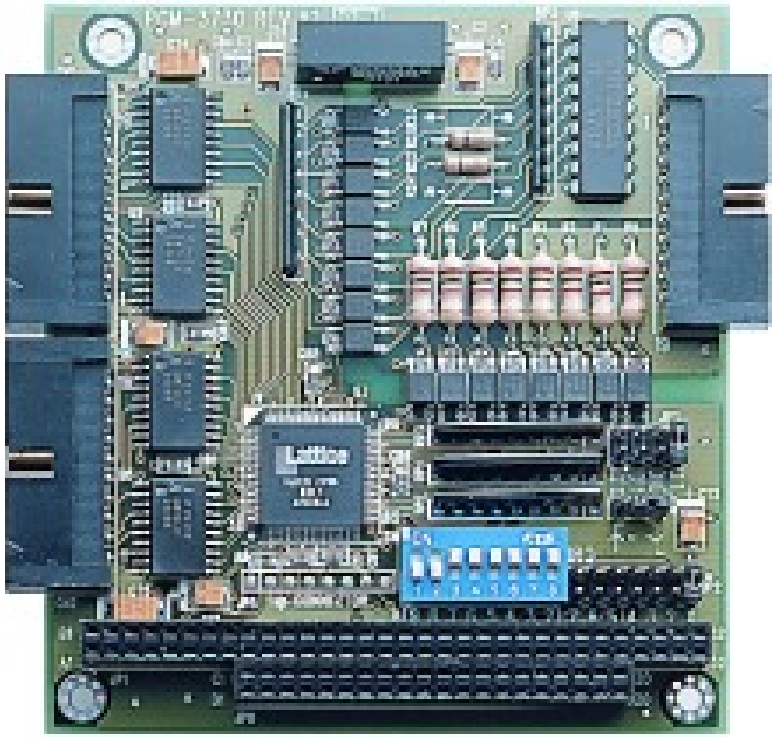
Konflikt nie powoduje opóźnienia transmisji ramki. Ten typ transmisji określa się jako CSMA/CD NDA (Carrier Sense Multiple Access/Collision Detection with Non-Destructive Arbitration)

Szybkość transmisji determinowana koniecznością wykrycia bitu recesywnego i prędkością światła. W przypadku długich odcinków potrzebne retransmitery (repeaters)

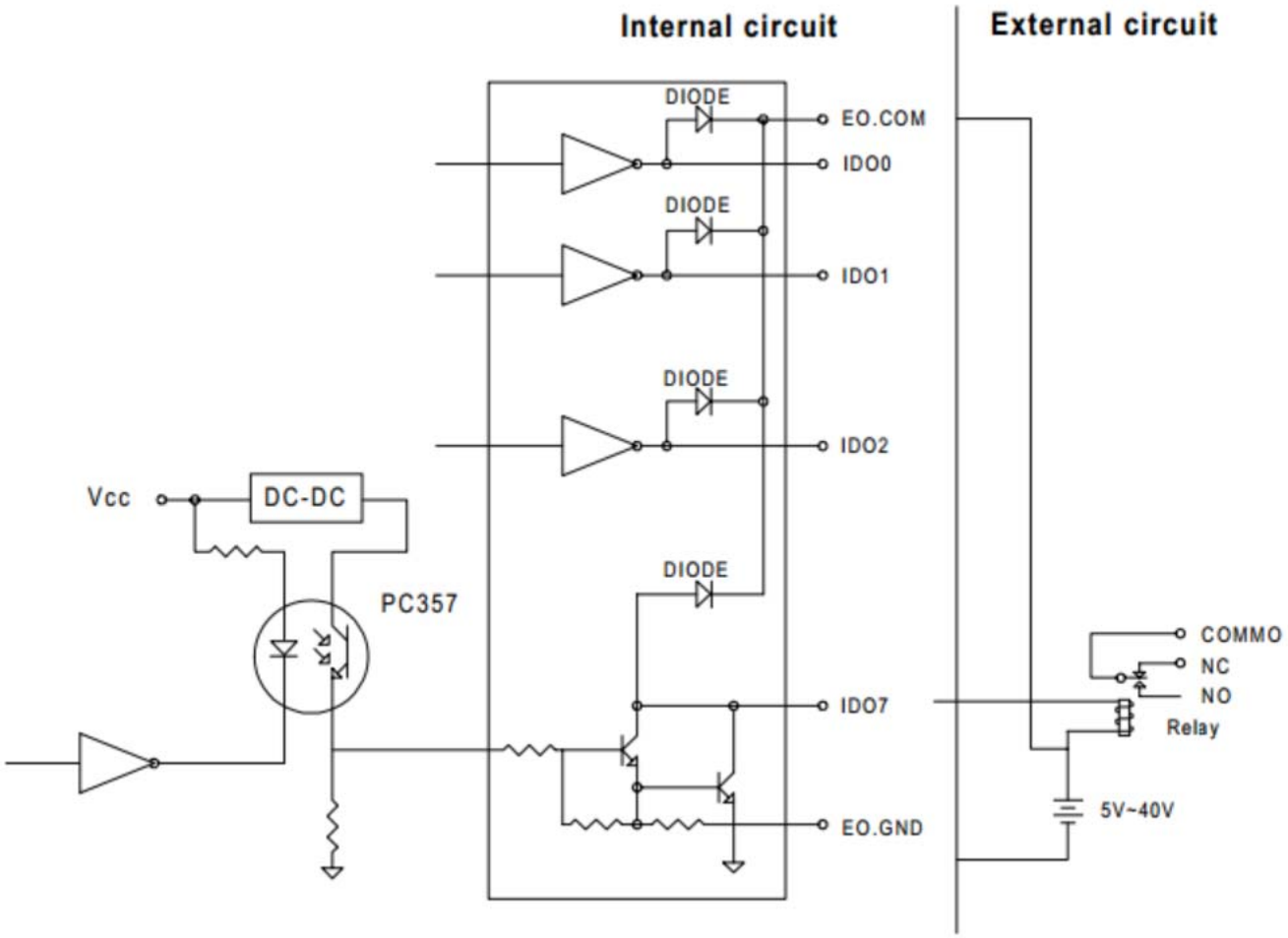
PCM-3730 [2]

- 8 wejść i 8 wyjść cyfrowych z optoizolacją
- 16 wejść i 16 wyjść TTL



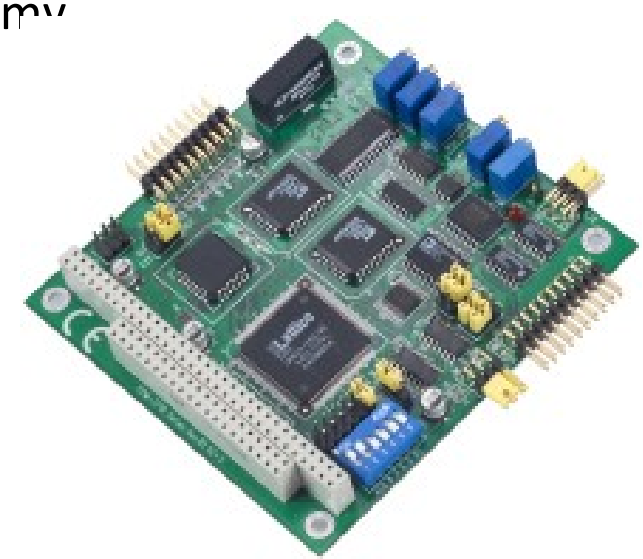


PCM-3730 [2]

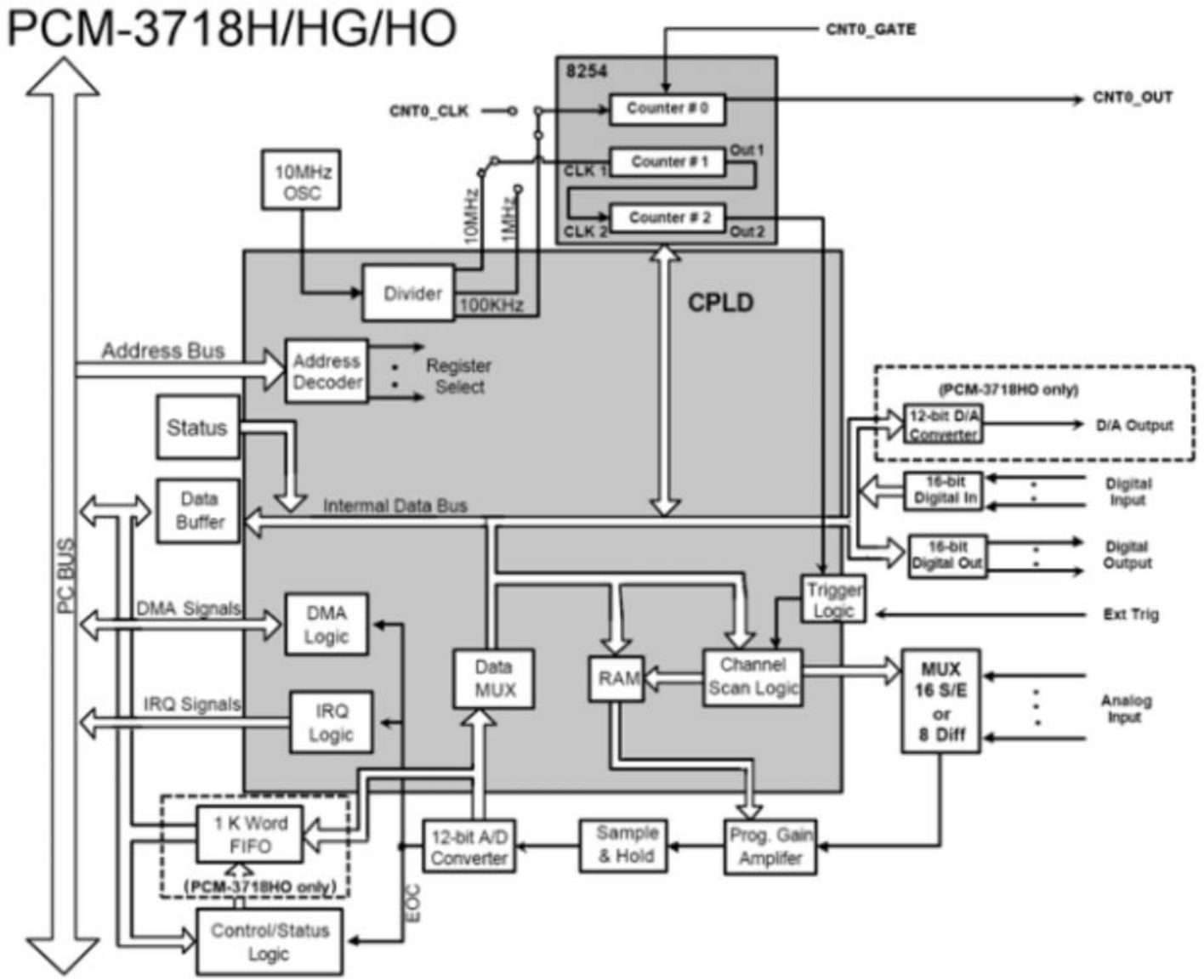

Internal circuit
External circuit
DIODE
EO.COM
IDO0
DIODE
IDO1
DIODE
IDO2
Vcc
DC-DC
PC357
DIODE
IDO7
EO.GND
COMMO
NC
NO
Relay
5V~40V

PCM-3718HO [2]

- 16 odniesionych do masy lub 8 różnicowych wejść analog
- 12-bit przetwornik A/D, do 100k próbek/s, współczynnik próbkowania równy DMA
- Dwa 8-bitowe cyfrowe wejścia/ wyjścia, poziomy TTL
- 12-bit wyjście analogowe



PCM-3718HO [2]


PCM-3718H/HG/HO
CNT0_GATE
8254
CNT0_CLK
Counter # 0
CNT0_OUT
10MHz OSC
Counter # 1
Out 1
CLK 1
10MHz
1MHz
Counter # 2
CLK 2
Out 2
Divider
100KHz
CPLD
Address Bus
Address Decoder
Register Select
(PCM-3718HO only)
12-bit D/A Converter
D/A Output
Status
16-bit Digital In
Digital Input
Data Buffer
Internal Data Bus
16-bit Digital Out
Digital Output
Trigger Logic
Ext Trig
PC BUS
DMA Signals
DMA Logic
Data MUX
RAM
Channel Scan Logic
MUX 16 S/E or 8 Diff
Analog Input
IRQ Signals
IRQ Logic
1 K Word FIFO
(PCM-3718HO only)
12-bit A/D Converter
Sample & Hold
Prog. Gain Amplifer
EOC
Control/Status Logic

Inne moduły

- Liczniki, moduły enkoderów
- Generatory PWM
- Sterowniki silników krokowych
- Szybkie przetworniki A/C
- Izolowane galwanicznie przetworniki A/C i C/A
- Cyfrowe wyjścia przekaźnikowe mechaniczne lub SSR (Solid State Relay) na 230 V AC

- Rekonfigurowalne moduły wyposażone w FPGA

Systemy operacyjne

Tryb pracy komputera w czasie rzeczywistym charakteryzuje się stałą gotowością programu przetwarzającego nadchodzące z zewnątrz dane. Oznacza to dostępność wyników przetwarzania w ściśle określonym przedziale czasu. W zależności od typu sterowanego obiektu czasy pojawienia się przetwarzanych danych mogą być przypadkowe lub ściśle określone.

Nie wystarcza poprawność obliczeniowa, obliczenia muszą być gotowe na czas

System RT to niekoniecznie system szybki, ale system zależny od upływającego czasu, odpowiednio lokujący zasoby aby spełnić wymagania czasowe

Soft real-time / hard real-time

System operacyjny – oprogramowanie zarządzające sprzętem oraz aplikacjami komputera (udostępnia zasoby aplikacjom)

System operacyjny czasu rzeczywistego (RTOS – real-time operating system) jest systemem operacyjnym gwarantującym określoną wydajność w ramach wyspecyfikowanych ograniczeń czasowych (gwarancja dostępności zasobów w zdefiniowanych limitach czasu).

Systemy operacyjne

RTOS: QNX, Windows CE, LynxOS, VxWorks, FreeRTOS, µC/OS, etc.

Modyfikacja Linux: RTLinux, RTAI

Sposoby zmiany Windows w system czasu rzeczywistego:

- dodać drugi komputer (wewnątrz lub na zewnątrz)
- przejąć przerwania NMI
- zaimplementować MS-Windows API w „prawdziwym” systemie czasu rzeczywistego
- uruchomić MS-Windows jako zadanie o niskim priorytecie w SO czasu rzeczywistego
- zmodyfikować HAL dodając małe jądro czasu rzeczywistego
- zainstalować małe jądro czasu rzeczywistego pracujące na poziomie 0 ochrony CPU (RTWT)

QNX

Mikrojądro (odmiennie od monolitycznych jąder wielu OS). Skalowalne – dodawanie nowych funkcji na zasadzie nowych modułów. Mikrojądro zawiera wyłącznie scheduler, IPC, obsługę przerwań i timery

IPC oparte o wysyłanie komunikatów i oczekiwanie na potwierdzenie. Wszystkie operacje w systemie oparte o ten mechanizm

Mikrojądro oraz dodatkowe moduły w naturalny sposób umożliwiają budowę systemów rozproszonych – jedyna modyfikacja to podróżujące sieciowo komunikaty

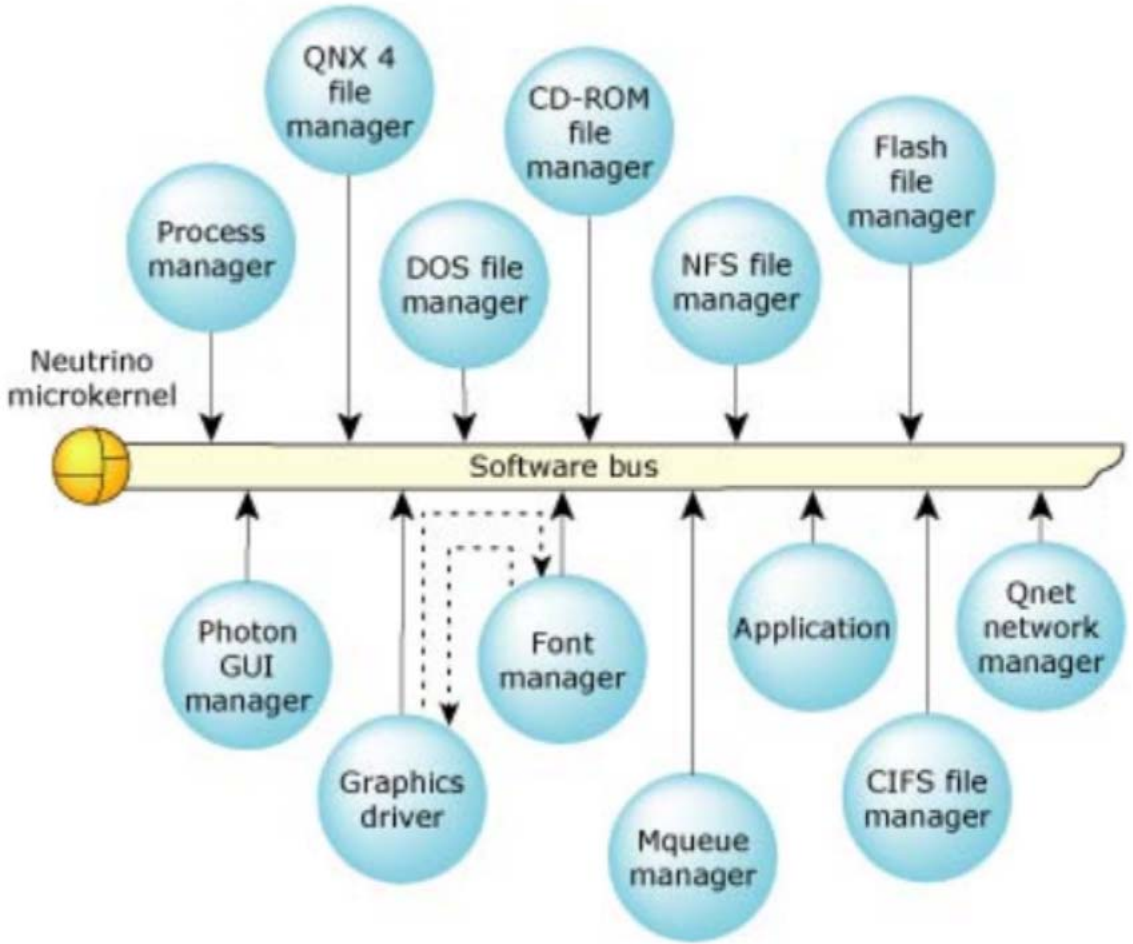
Skalowalność umożliwia zastosowania zarówno typu embedded jak i w dużych rozproszonych systemach

Dostępne na wiele platform, np. x86, ARM

Dostępna bezpłatna wersja systemu dla zastosowań niekomercyjnych

Uważany za jeden z najsilniejszych i najbardziej stabilnych RTOS na rynku

QNX


QNX 4
file
manager
CD-ROM
file
manager
Flash
file
manager
Process
manager
DOS file
manager
NFS file
manager
Neutrino
microkernel
Software bus
Photon
GUI
manager
Font
manager
Application
Qnet
network
manager
Graphics
driver
Mqueue
manager
CIFS file
manager

FreeRTOS

BEZPŁATNY, dostępne kody źródłowe

Wspiera 33 procesory, 77000 pobrań rocznie

Skalowalny, przeznaczony do niewielkich systemów embedded. Jądro systemu ma typowo 4-9kB

Wspiera pracę w trybie wywłaszczającym

Posiada tryb pracy z oszczędzaniem energii dla systemów zasilanych bateryjnie

Mechanizmy IPC: kolejki, semafory, mutex. Efektywny mechanizm timerów

Bogaty zestaw narzędzi developerskich

Narzędzia cross-development dla środowiska Windows

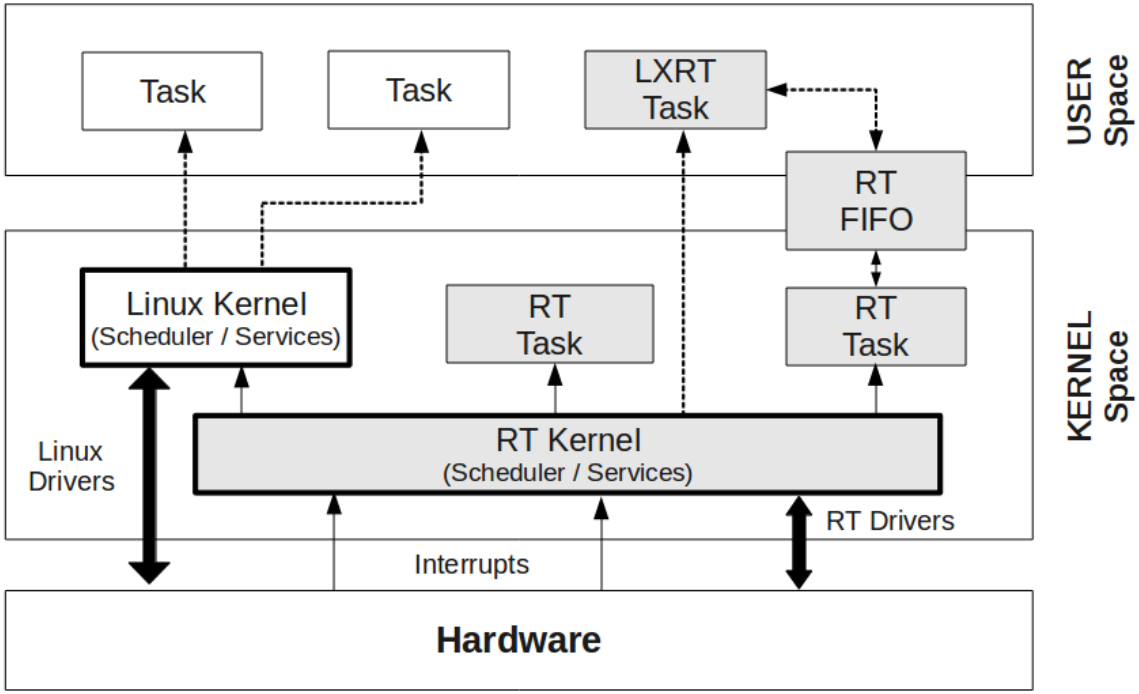
RTAI Real-time Application Interface

Bezpłatne rozszerzenia Linux o funkcje czasu rzeczywistego

Oferuje zestaw funkcji API dedykowany do aplikacji RT

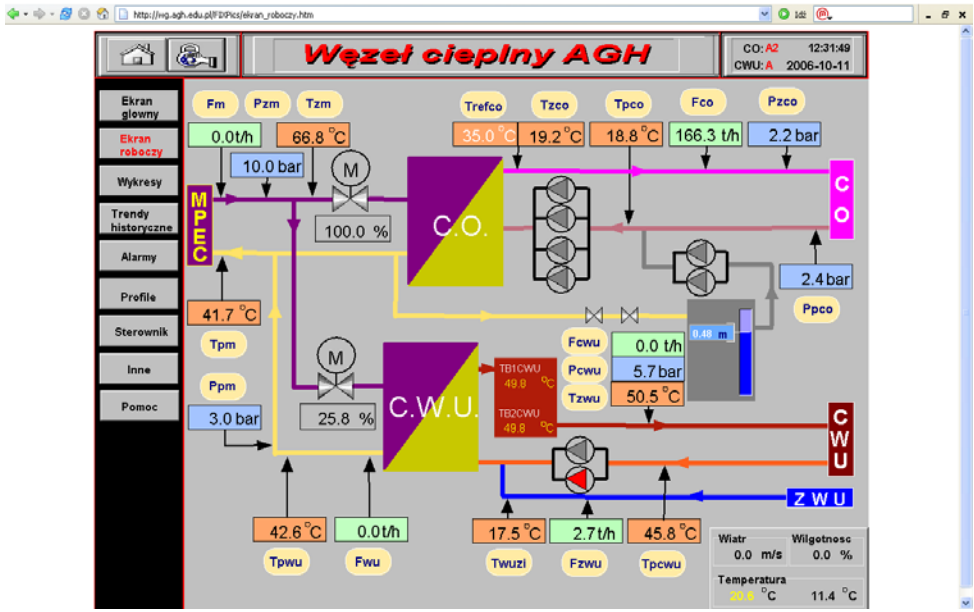
Dostępne instalacje dla różnych dystrybucji Linux

Wspiera różne platformy sprzętowe np. x86, ARM, PowerPC



SCADA/HMI Supervisory Control and Data Acquisition

- sygnałowa baza danych,
- **tworzenie/wyświetlanie ekranów synoptycznych,**
- system alarmów,
- trendy historyczne,
- system bezpieczeństwa,
- praca sieciowa,
- sterowniki urządzeń,
- serwery DDE /OPC,
- wspieranie ODBC / OLE DB,
- VBA oraz skryptowy język komend.



Platformy sprzętowe

SCADA

PLC

Proces

Czujniki i elementy wykonawcze

Real-time information portal

Plant performance

Historian

HMI / SCADA

Control / IO | Control / IO | Control / IO

PLC

Realizują w przemyśle większość sterowania bezpośredniego

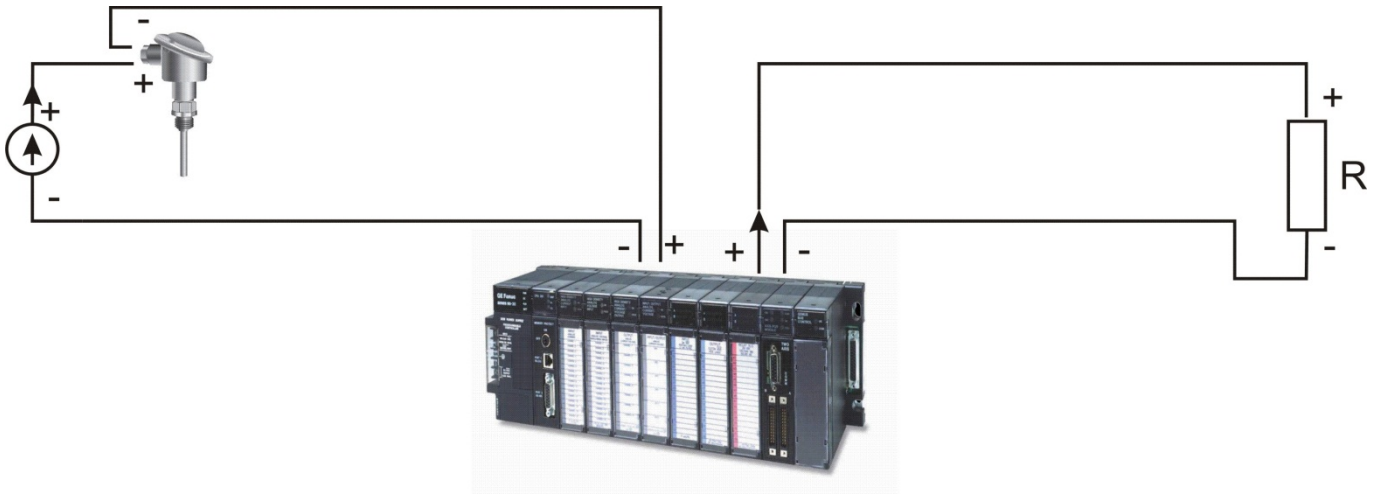
Modułowa budowa: zasilacz, CPU + moduły I/O i komunikacyjne

Zasadą pracy cykliczne wykonywanie programu

Języki programowania (IEC 61131-3):

- **LD** (*Ladder Diagram*)
- **FBD** (*Function Block Diagram*)
- **ST** (*Structured Text*)
- **IL** (*Instruction List*)
- **SFC** (*Sequential Function Chart*)

PLC pętla prądowa 0-20mA / 4-20mA



Zasilanie czujników z pętli prądowej

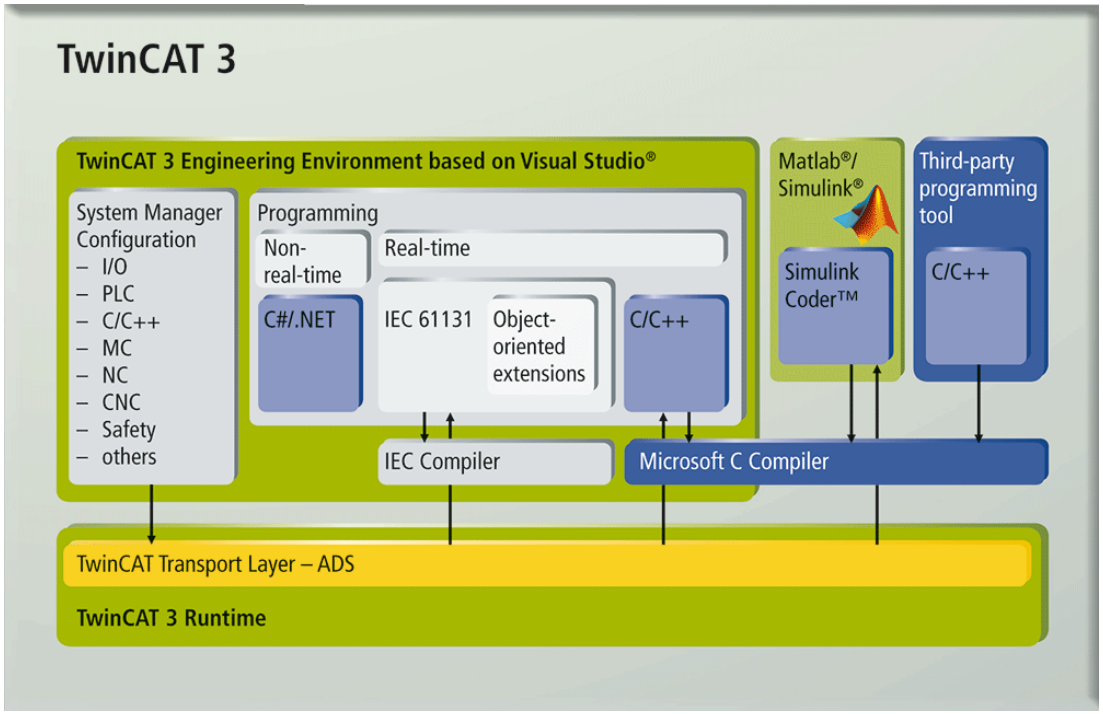
Odporność na zakłócenia: transmisja „skrętką” oraz niskie rezystancje w pętli prądowej

Soft PLC

Oprogramowanie pracujące na komputerze (przemysłowym) realizujące funkcje sterownika PLC

Oprogramowanie realizuje możliwość programowania w każdym z języków IEC 61131-3

Np. TwinCAT firmy BeckHoff pracujący pod kontrolą WinCE



Embedded system

System zanurzony (ang. embedded) – dedykowany system komputerowy przeznaczony do wykonania jednego konkretnego zadania; to połączenie sprzętu oraz oprogramowania (często opartego o RTOS) ściśle ukierunkowane na konkretną aplikację

Zdecydowanie ukierunkowane na wykonanie konkretnego zadania, a nie na bycie uniwersalnym

Przykłady: odtwarzacz MP3, mapa GPS, sterowanie kuchenką mikrofalową, sterowanie silnikiem spalinowym, pilot samochodowy, itp. Znaczna część systemów embedded związana jest ze sterowaniem

Do budowy wykorzystywane mikrokontrolery oraz procesory DSP

Często urządzenia niewielkie (w sensie złożoności układów elektronicznych) – brak standardów budowy

Zastosowanie systemu operacyjnego procentuje podczas rozwoju systemu sterowania

Mikrokontrolery

Układ μC zawiera:

- CPU
- Pamięć danych i programu
- Układy we/wy (liczniki, zegary, we/wy szeregowe, we/wy równoległe, sterownik przerwań, przetworniki A/C i C/A, generatory PWM, interfejsy do enkoderów, bloki CAN, itp.)

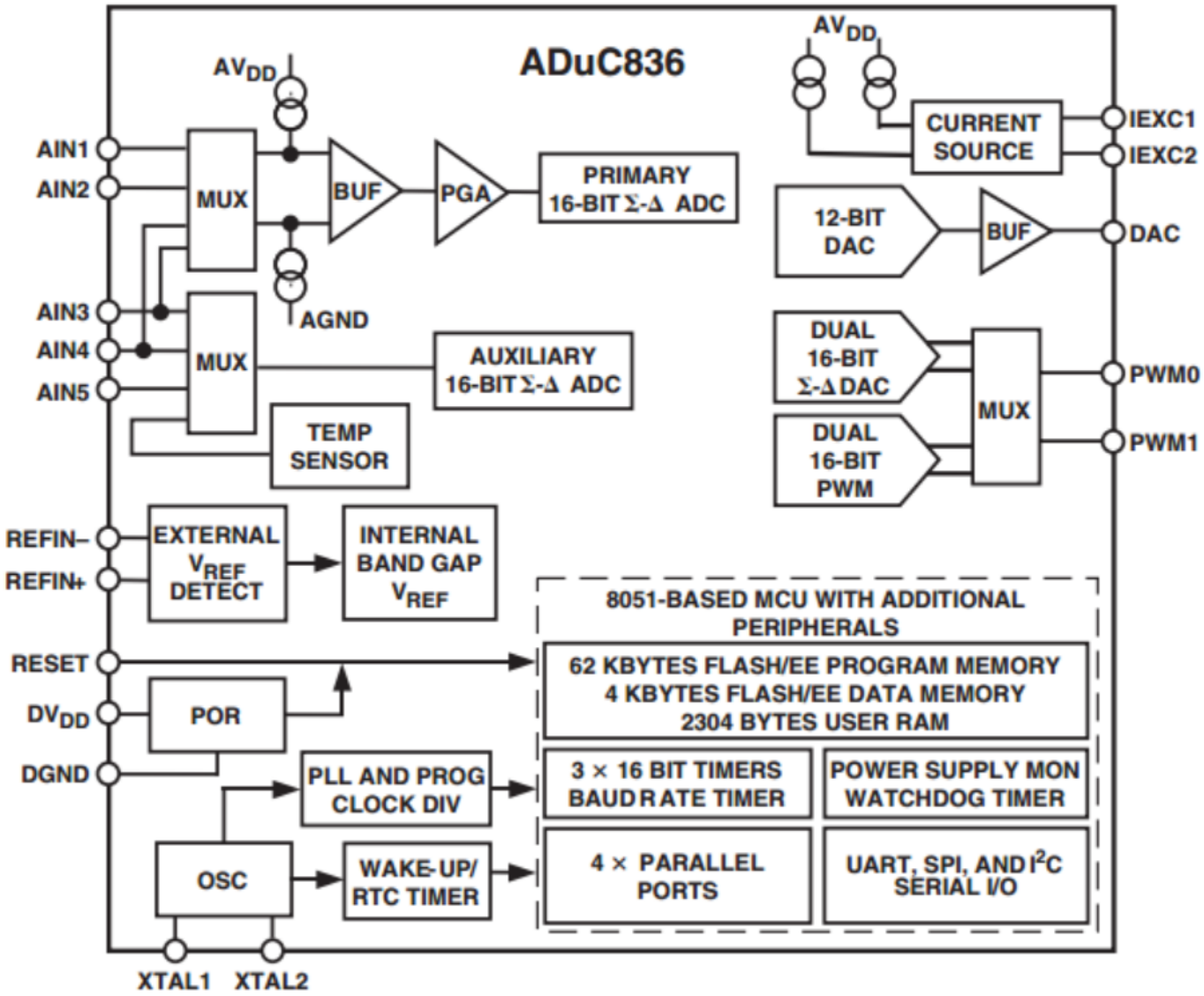
Silna integracja

Specjalizowana architektura

Niski koszt w związku ze zintegrowanymi peryferiami

Różnorodne wykonania: różna liczba i rodzaj peryferii, rozmiar pamięci, liczba wyprowadzeń, wartość napięcia zasilania, szybkość zegara, pobór mocy. **Należy spróbować dobrać typ do aplikacji.**

Platformy sprzętowe



Sieci przemysłowe

Komunikacja czasu rzeczywistego – tryb pracy, w którym wszystkie węzły mogą wymieniać informację ciągle lub z pomijalnymi (akceptowalnymi i ściśle określonymi) opóźnieniami lub

Sieć umożliwiająca pracę systemów komputerowych w czasie rzeczywistym lub

Sieć gwarantująca wykonanie usług, a więc deterministyczna

Systemy komputerowe aplikacji sterujących stają się coraz bardziej zaawansowane oraz powszechne, stąd naturalna tendencja do budowy architektur rozproszonych, pracujących zgdnie z wymogami czasu rzeczywistego

Wykorzystywano dedykowane sieci: np. Modbus, Profibus, DeviceNet, CANopen. Wraz ze wzrostem popularności Ethernet pojawiła się presja wykorzystania tej sieci w systemach sterowania

Ethernet

Żaden z wymienionych standardów nie jest tak popularny i nie ma takiego pasma jak Ethernet. 1Gbps jest około 100x szybszy od ProfiBus

Komunikacja w sieci Ethernet (IEEE 802.3) jest NIEDETERMINISTYCZNA.

CSMA/CD (Carier Sense Mulptiple Access/Collision Detection) – węzeł nasłuchuje medium komunikacyjne. Jeżeli nie ma transmisji rozpoczyna nadawanie, ale nadawanie może rozpocząć kilka węzłów. Węzły podczas nadawania nasłuchują i o ile wykryją kolizje transmitowane ramki są tracone, a węzły się odłaczają

Kolizja może być wykryta w pewnym oknie czasowym na początku transmisji. Długość okna czasowego determinuje maksymalną długość nośnika

Ethernet

Algorytm odłączania po kolizji (backoff algorithm):

- Początkowo n=0
- Rozpoczynając transmisję wykonuje się n++ - jest to licznik kolizji
- Jeżeli n>16 kończy się próby transmisji i powiadamia wyższe warstwy protokołu
- Dla n<=16 wylicza się k=min(n,10)
- Losuje się liczbę r z zakresu $0…2^k$
- Oczekuje się r okien czasowych z próbą wznowienia transmisji (okno czasowe to 5µs dla 100Mbps)

Możliwość całkowitego niewysłania pakietu lub wysłania w czasie przypadkowym

Ethernet

Ethernet jest niedeterministyczny, ale tylko gdy występują kolizje. Usunięcie kolizji uczyni z niego sieć RT

Metody usuwania kolizji:

•Zastosowanie sprzętu sieciowego w postaci urządzeń *Switch* – zestawiają chwilowe połączenia między nadawcą i odbiorcą

•PROFInet – wykorzystuje zmodyfikowany stos TCP/IP; wydzielony programowo kanał komunikacji RT – czasy reakcji 5-10ms

•RROFInet-IRT (Isochronous RT) – wspomagany sprzętowo; dzieli cykl komunikacyjny na RT i non-RT. Okna dla obu typów komunikacji synchronizowane IEE 1588 (PTP –Precision Time Protocol); reakcja 1ms

•EtherCAT – wspomagany sprzętowo (układy ASIC); zasada master-slave – master nadzoruje komunikację, a slave odpowiada wyłącznie na żądanie; w jednej ramce Ethernet komendy dla wielu urządzeń; architektura typu *ring* umożliwia dodawanie do przepływającego telegramu danych w trakcie ruchu telegramu; 1000 I/O w 30us - 200 analogowych I/O w 50us – 100 osi servo w 100us

Realizacja funkcji we/wy:

- PROGRAMOWO w oparciu o „uniwersalne” we/wy - wolna
- SPRZĘTOWO w oparciu o dedykowane układy scalone – szybkie ale nieelastyczne

Przykład wejść enkodera – program może obsłużyć ręczny zadajnik enkoderowy; enkoder pomiarowy o dużej rozdzielczości musi być obsługiwany sprzętowo

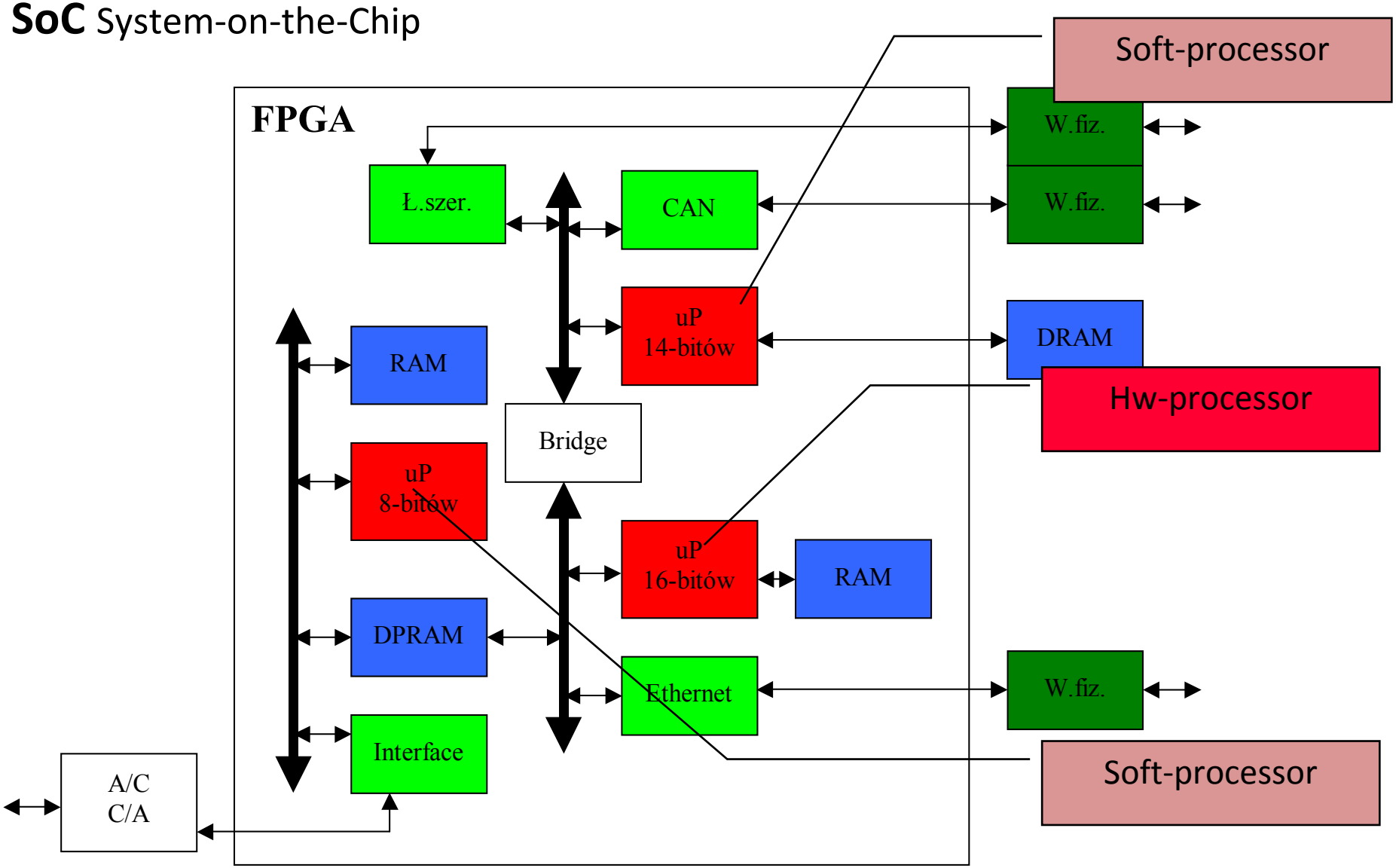
Następstwa:

- PRZEWYMIAROWANIE” – instalacja większej od potrzeb liczby we/wy
- Brak elastyczność – dedykowane do konkretnej instalacji (kosztowne zmiany !!!)

Rozwiązanie: zaprojektować sobie własne moduły we/wy

Zastosujmy COŚ, gdzie typ we/wy będzie można zmieniać

SoC System-on-the-Chip



FPGA

FPGA (Field Programmagle Gate Array) to:

- zestaw sekwencyjnych oraz kombinacyjnych elementów logicznych (tj. przerzutników i bramek)
- zestaw bloków I/O definiujących funkcje wyprowadzeń układu scalonego
- zestaw elementów połączeniowych łączących ww. elementy

Pojemności układów do 8 milionów bramek (8-bitowy mikrokontroler wymaga tysięcy bramek), liczba wyprowadzeń po wykorzystania przez użytkownika ~1500, czas propagacji pojedynczej komórki <1ns

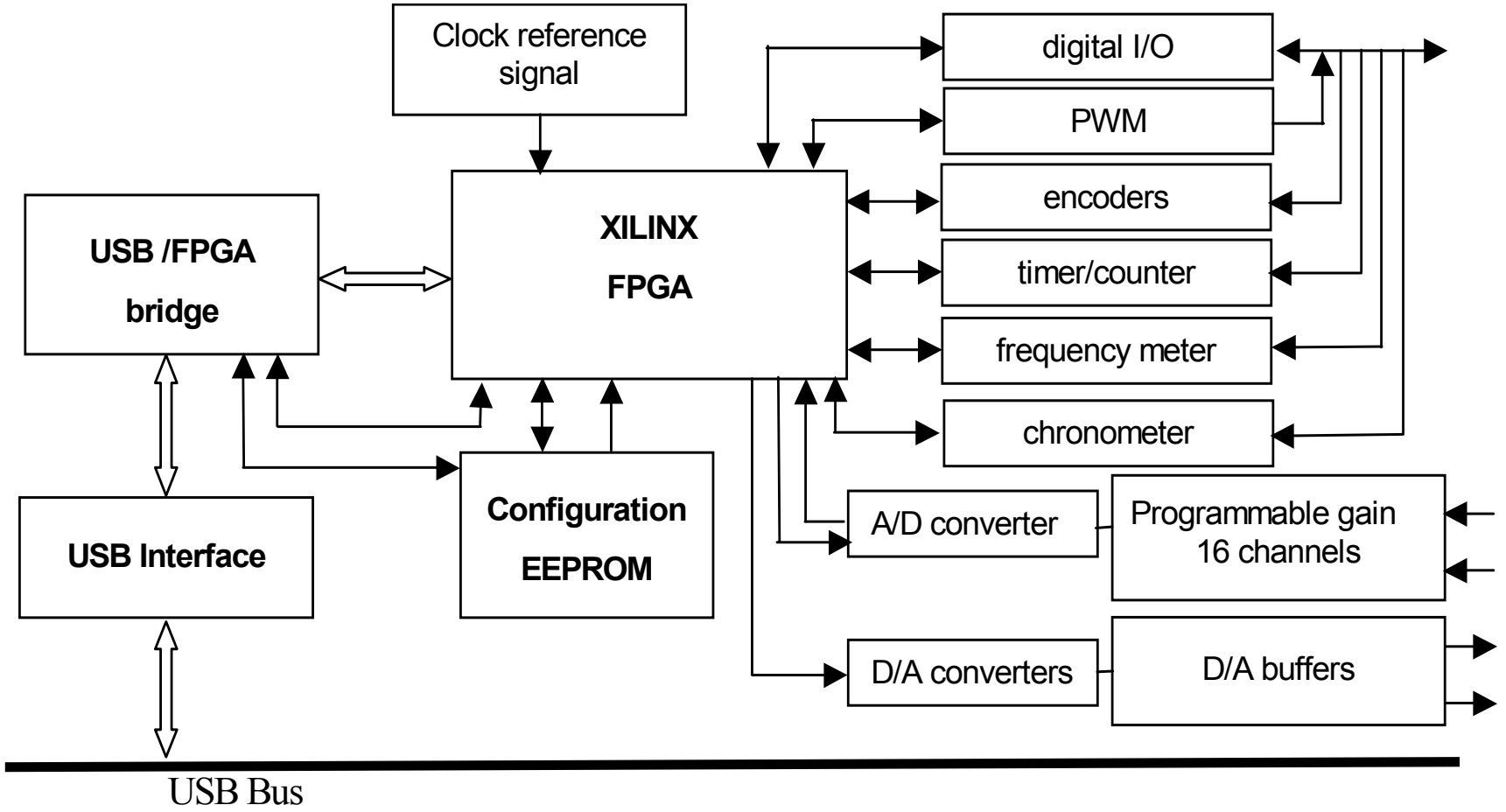
Funkcje elementów składowych układu FPGA nie są określone przez producenta lecz przez użytkownika układu

Programowanie w językach opisu sprzętu (VHDL)

Możliwość wielokrotnego określenia funkcji układu; zmiana funkcji układu nie wymaga zmian sprzętowych

FPGA umożliwia tworzenie przez użytkownika cyfrowych układów scalonych o wymaganych funkcjach realizowanych SPRZĘTOWO.

FPGA


Clock reference signal
digital I/O
PWM
encoders
timer/counter
frequency meter
chronometer
XILINX FPGA
USB /FPGA bridge
Configuration EEPROM
USB Interface
A/D converter
Programmable gain 16 channels
D/A converters
D/A buffers
USB Bus

Zynq Extensible Processing Platform (EPP) :

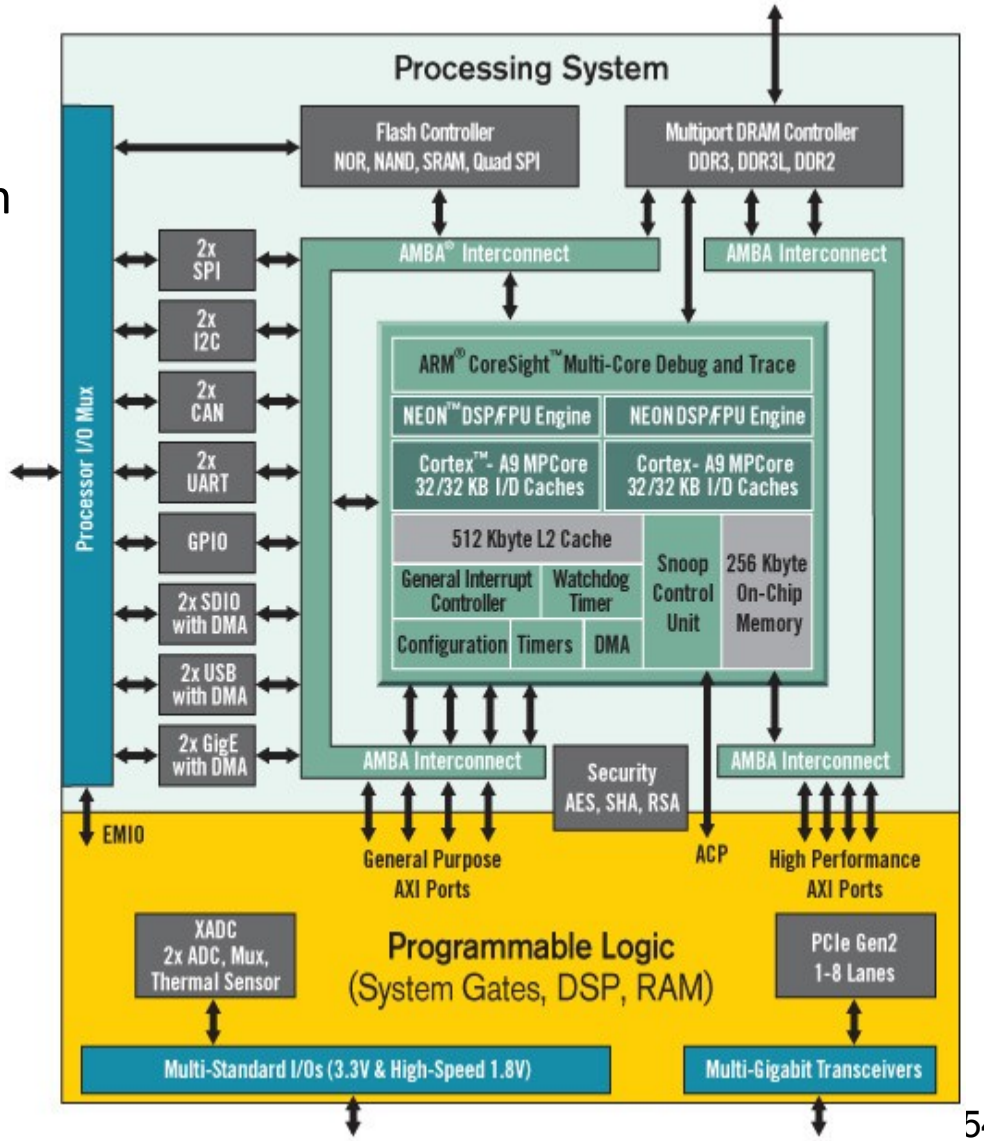
- Dostępne od czerwca 2012
- Dwukorowy ARM Cortex A9 1GHz z FPU
- Praca w konfiguracji AMP i SMP
- Linux, systemy operacyjne czasu rzeczywistego (QNX, VxWorks, FreeRTOS)
- Interface dla DDR i Flash memory
- Wbudowane interfejsy: USB 2.0, Ethernet, CAN, SD, UART, SPI, I2C, and GPIOs
- Dwa A/D converters (17 multiplexed inputs)
- FPGA do 6 milionów bramek

Zynq Extensible Processing Platform (EPP)

Dwukorowy system µP w pojedynczym IC (wymaga zewnętrznego RAM i pamięci masowej)

Zalety:

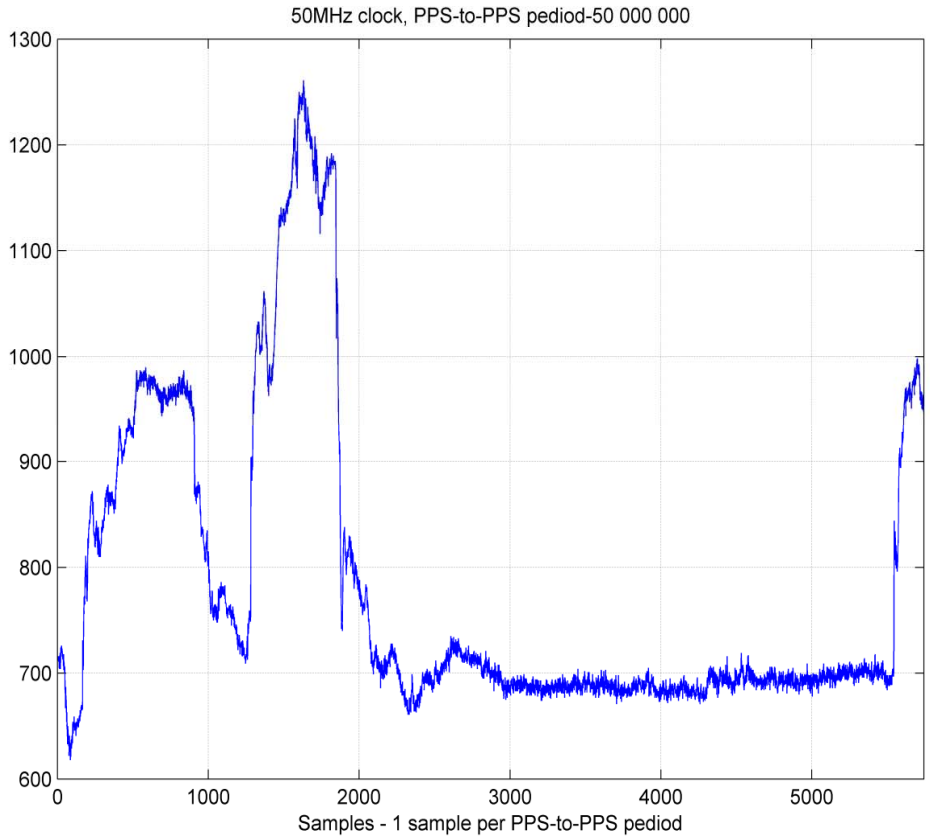
- Zmiany sprzętowo realizowanych funkcji
- Przesuwanie funkcji między realizacją sprzętową i programową



Przykład: precyzyjne odtwarzanie stempli czasowych z sygnału GPS

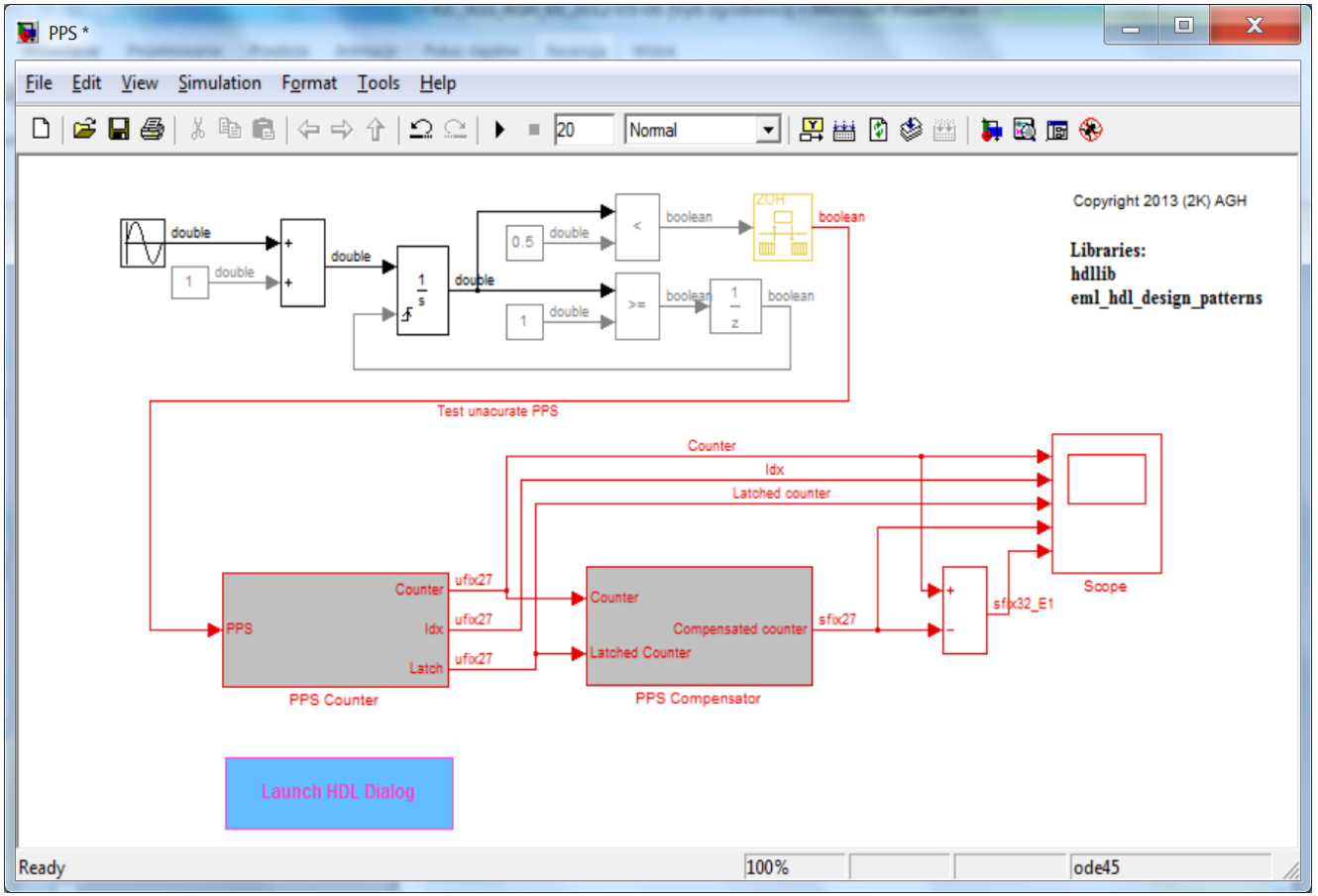
GPS Pulse Per Second (PPS) signal – dokładność 100ns-1µs

Dryft generatora kwarcowego- ~25µs / ~20^{o}C



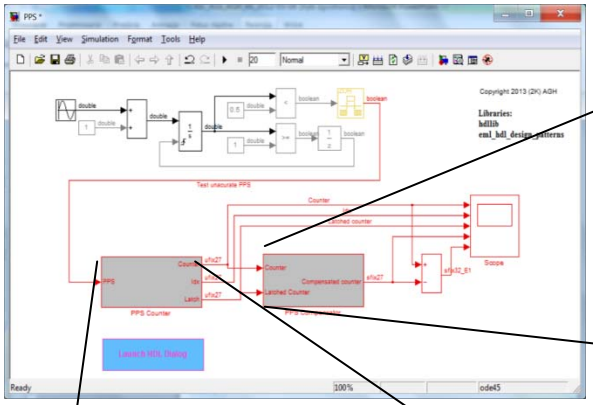
Przykład: precyzyjne odtwarzanie stempli czasowych z sygnału GPS

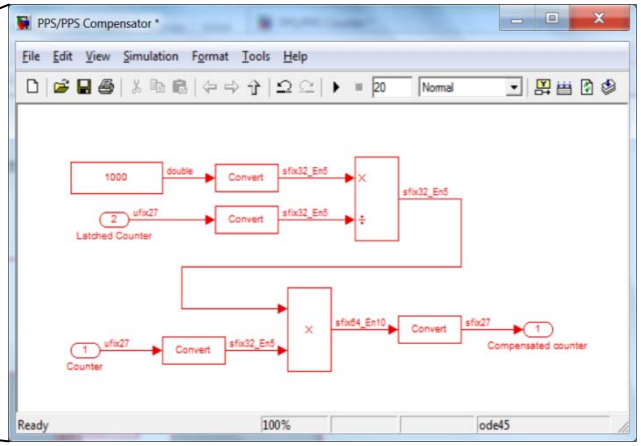
Kompensacja generatora kwarcowego

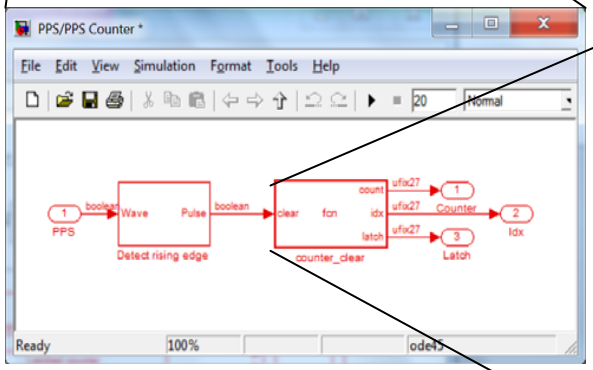


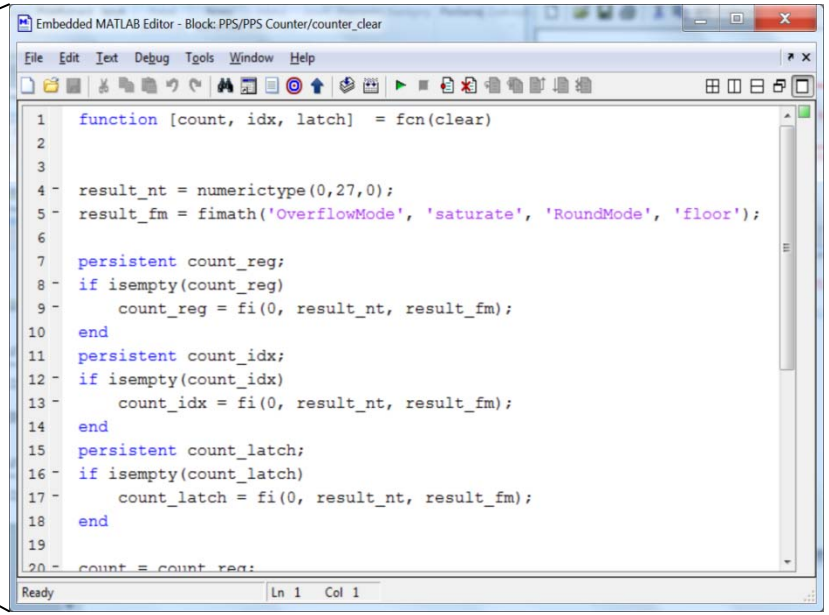
Przykład: precyzyjne odtwarzanie stempli czasowych z sygnału GPS

Kompensacja generatora kwarcowego

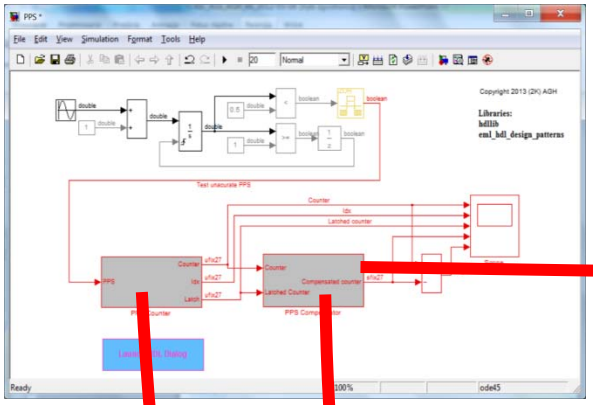


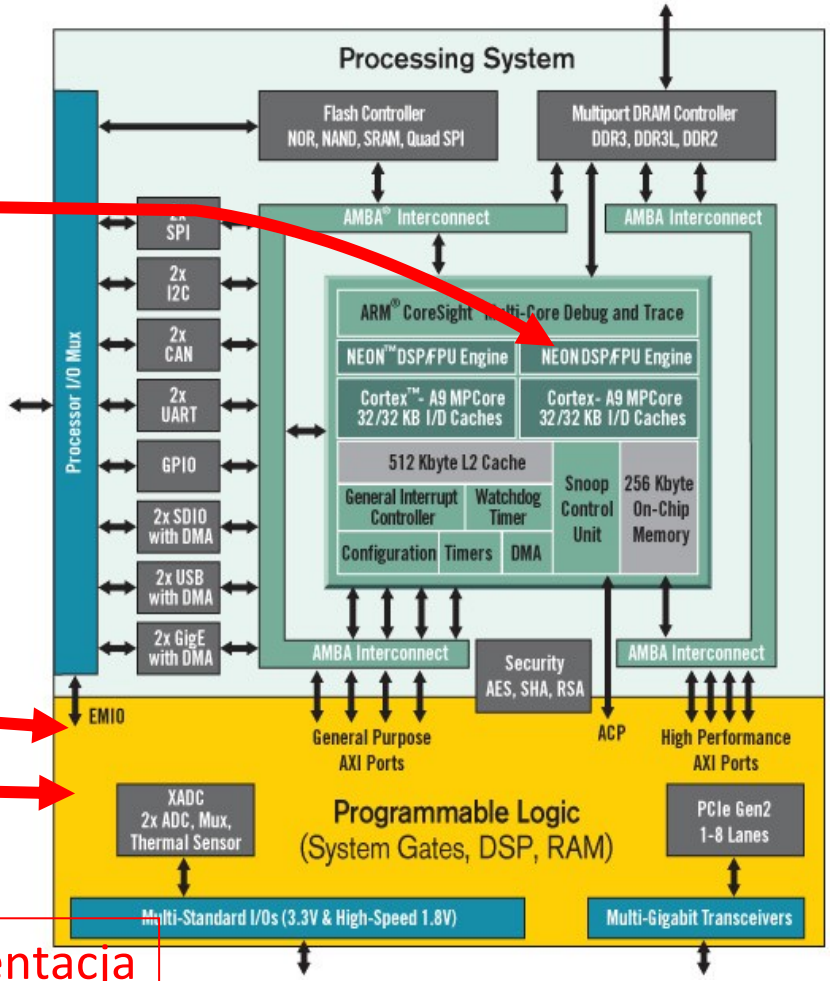






Przykład: precyzyjne odtwarzanie stempli czasowych z sygnału GPS

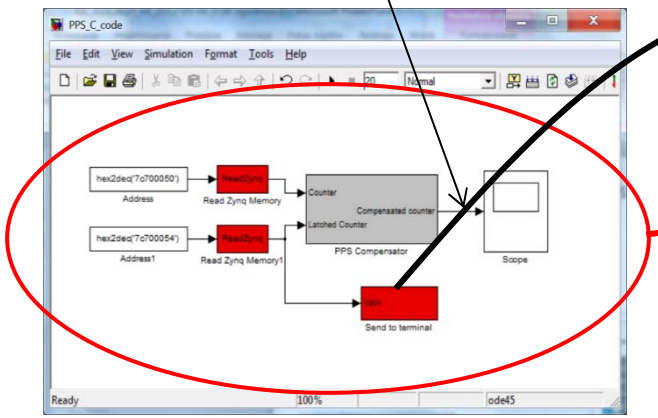


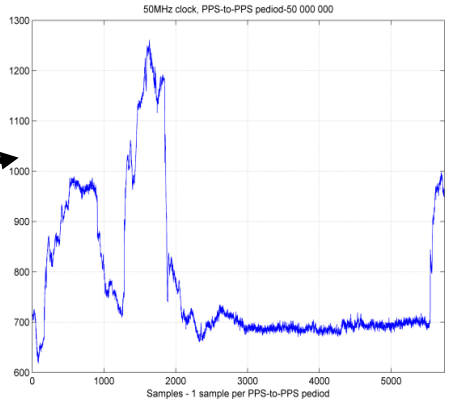


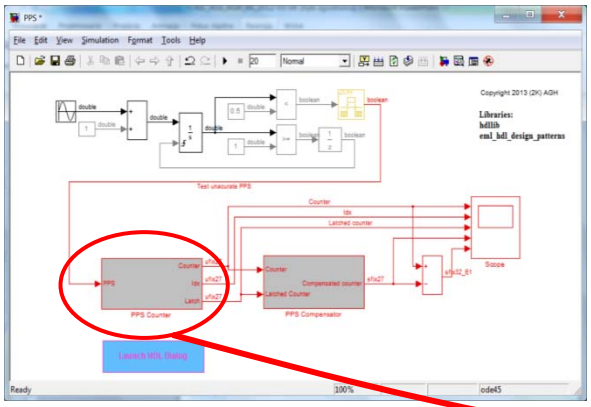
Automatyczna generacja kodu i implementacja

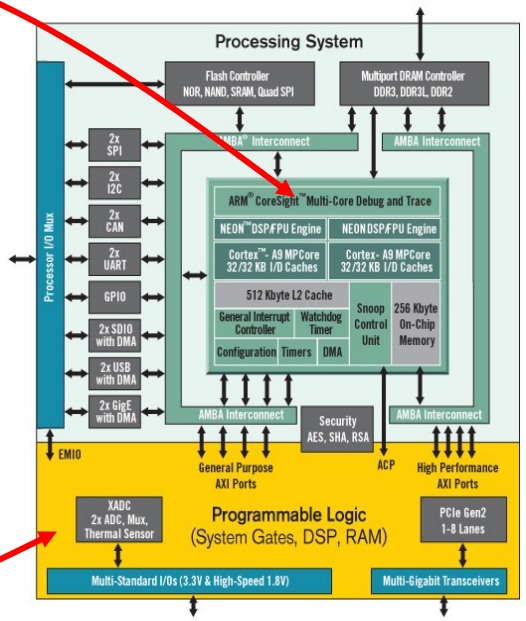
Przykład: precyzyjne odtwarzanie stempli czasowych z sygnału GPS

Estymowany precyzyjny stempel czasowy

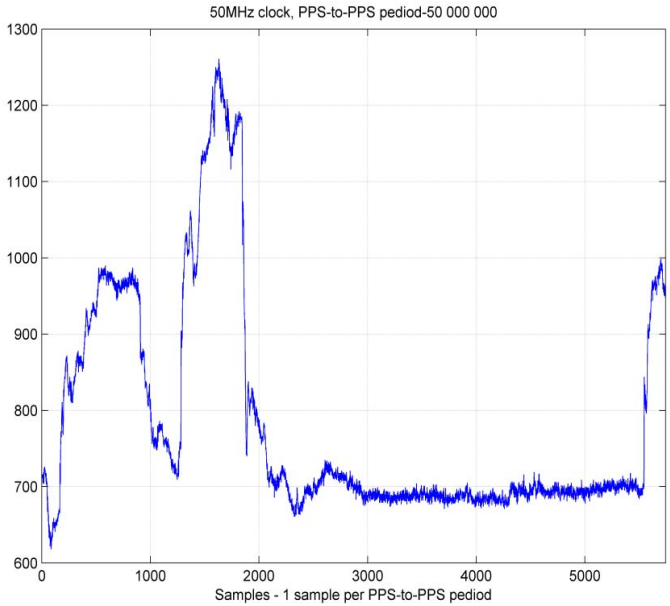


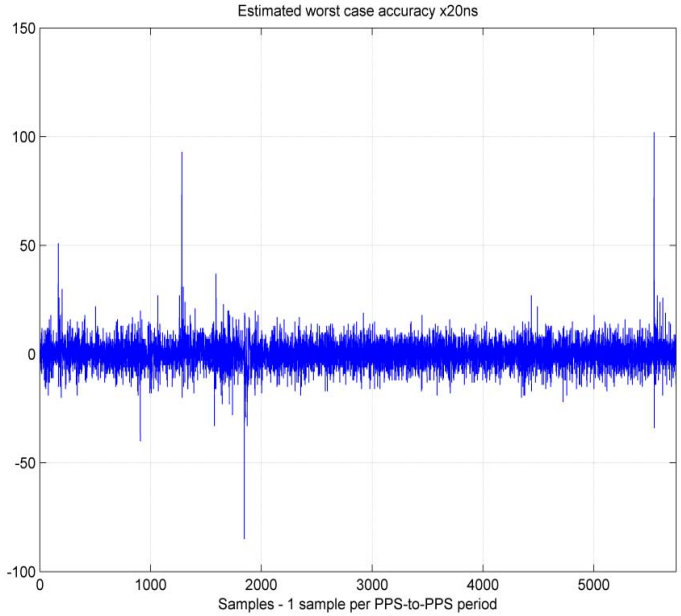






Przykład: precyzyjne odtwarzanie stempli czasowych z sygnału GPS





Dokładność: 25µs a 2µs

Utworzone w MATLAB/Simulink
(wymagane oprogramowanie XILINX)

Modbus

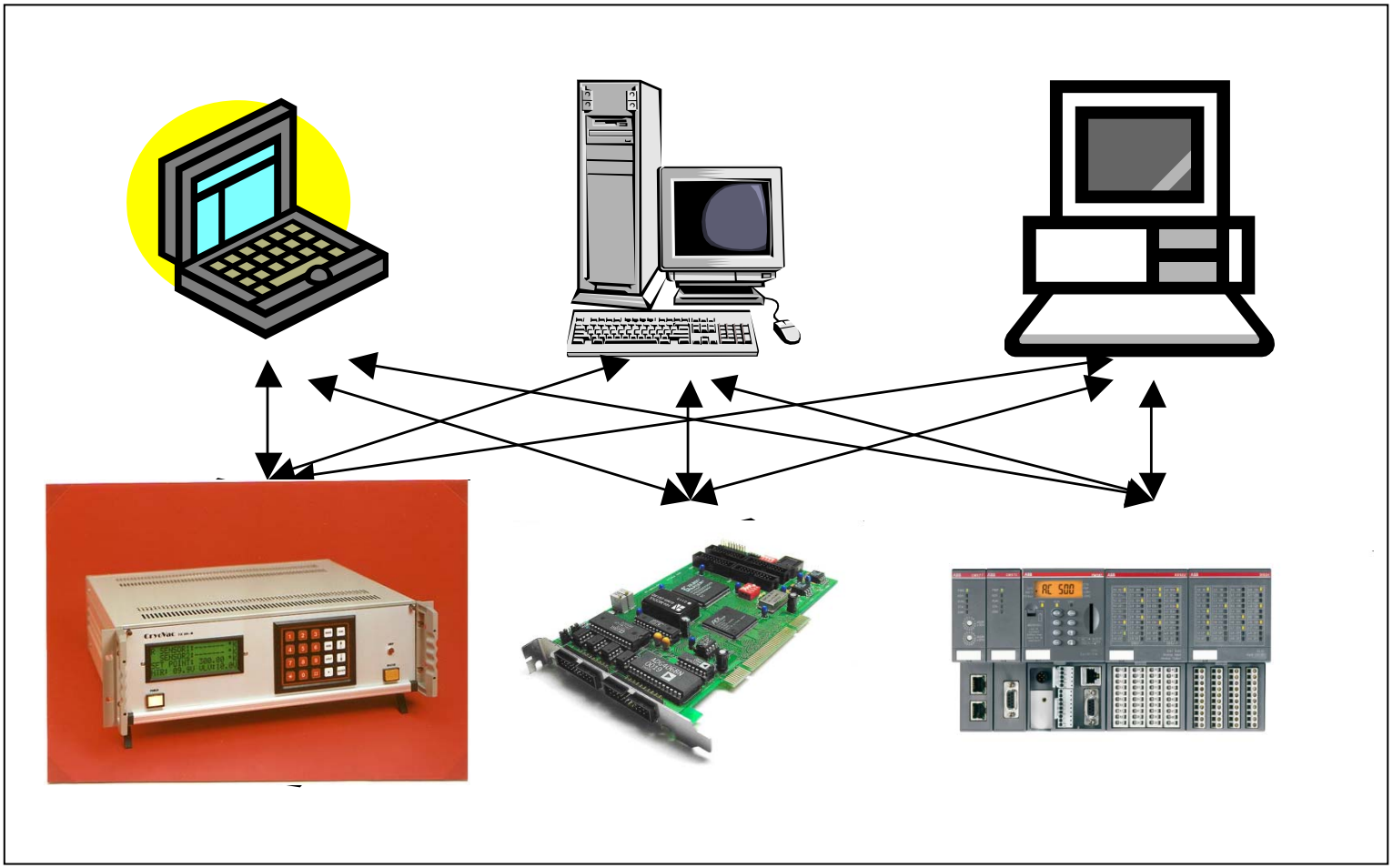
- Modbus ASCII – ramki w postaci znaków w kodzie ASCII
- Modbus RTU – ramki binarne

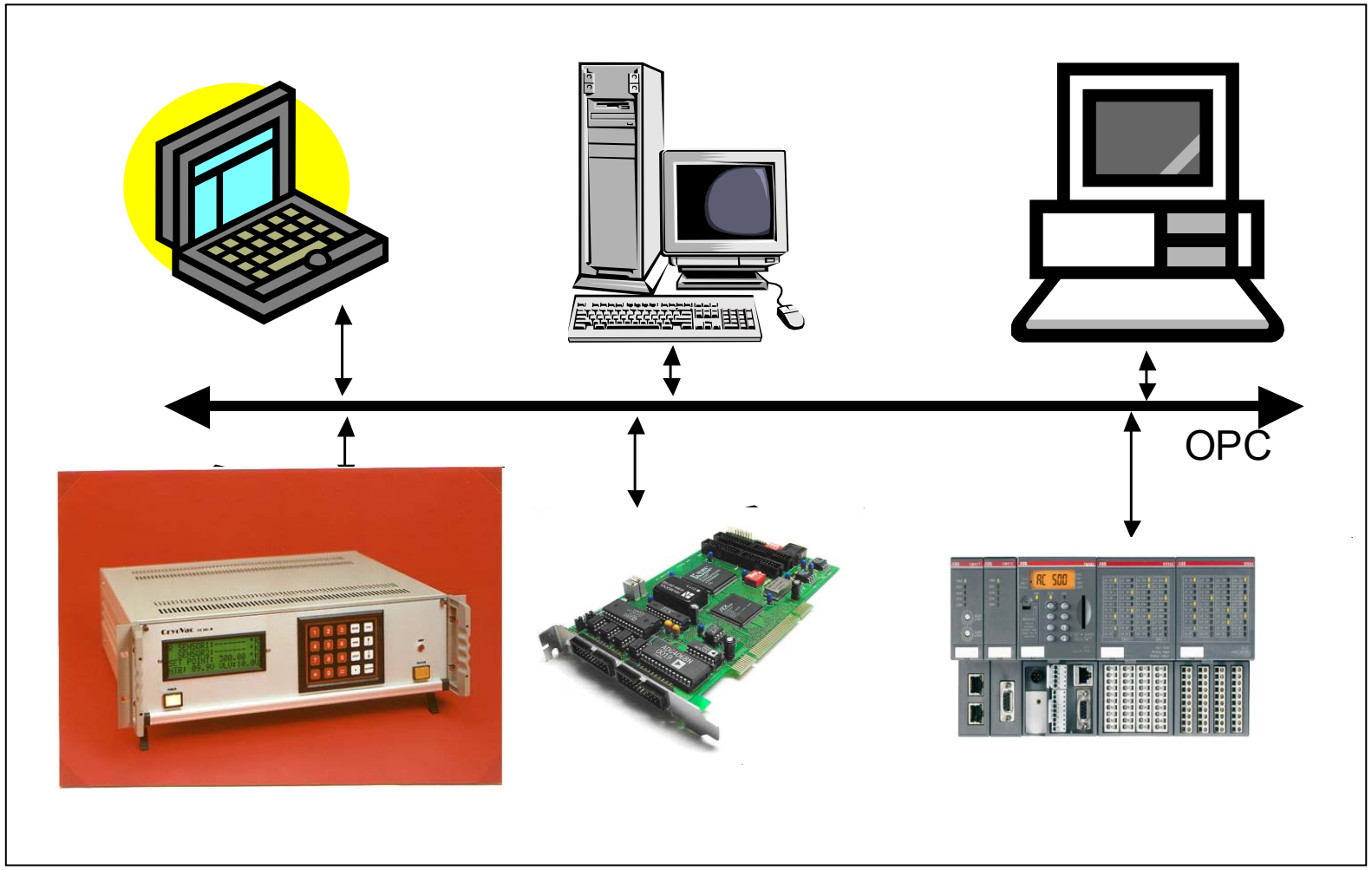
- Modbus TCP – Modbus RTU transmitowany poprzez TCP/IP
- **Niezwykle prosta implementacja**
- Zapis/odczyt danych o charakterze referencji sterowników PLC

Ethernet / TCP/IP

- Zapewnia łączność sieciową między dowolnymi systemami komputerowymi, poprzez różnorodne media sieciowe
- TCP/IP – podstawowe protokoły Internetu
- Zapewnia łączność dla różnych charakterystyk sieci: różnego pasma, opóźnień, utraty pakietów, zmiany kolejności pakietów, zmiennego rozmiaru pakietów, uszkodzeń fragmentów sieci
- **Niezwykle prosta implementacja** – np. biblioteka BSD Sockets ukrywa szczegóły implementacyjne TCP, UDP i IP
- Używane w warstwie aplikacji wielu programów

Komunikacja niezestandaryzowana





Potrzeba standardu

OPC (open connectivity via open standards**)**

(OLE for Process Control) (Openness Productivity Collaboration)

- **OPC – standard umożliwiający wygodny i efektywny sposób połączenia sprzętowych i programowych elementów automatyki**
- Zarządzany przez międzynarodową organizację (www.opcfoundation.org). Członkami wszyscy liczący się producenci urządzeń oraz oprogramowania dla automatyki
- OPC Foundation adaptuje istniejące technologie o otwartym charakterze (**nie tworzy** nowych) dla potrzeb automatyki. OPC bazuje na DCOM
- OPC Specification 1.0 pojawiło się w 1996. Powstało OPC Foundation.
- Dostępne specyfikacje:
  - **Data Access Specification** – mechanizmy odczytu i zapisu danych procesowych czasu rzeczywistego (danych aktualnych). Ciągła komunikacja wyrządzeń pomiarowo wykonawczych, PLC i DCS z oprogramowaniem do bieżącej prezentacji (HMI SCADA)
  - Alarms and Events Specification – monitorowanie i przetwarzanie alarmów i zdarzeń. Strumień danych nie ma charakteru ciągłego

OPC

- Historical Data Access Specification – udostępnianie danych historycznych (danych które zostały już zapisane)
- OPC Data eXchange – wymiana danych typu serwer-serwer, a nie klient-serwer
- OPC Security Specification – strategie bezpieczeństwa w komponentach; ochrona przed nieautoryzowanym dostępem lub modyfikację danych
- OPC Batch Specification – zarządzanie produkcją wsadową
- OPC and XML – wykorzystanie XML w komponentach OPC

... i ciągle trwają prace nad nowymi

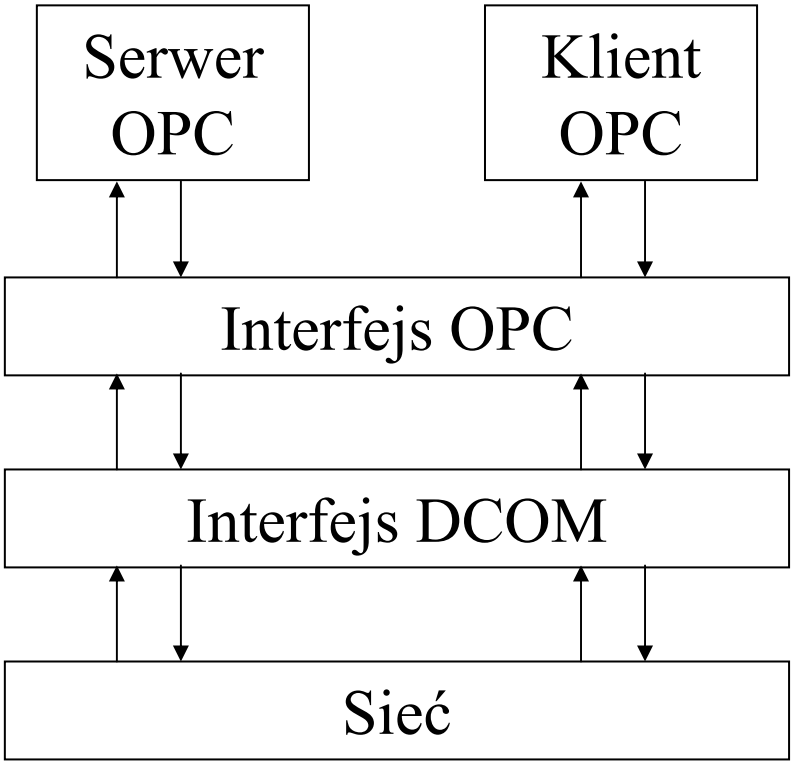
OPC

OPC – interoperatywność (zdolność wspólnego działania, ang. interoperability) dzięki standardowi komunikacji

OPC – umożliwia uniezależnienie oprogramowania (SCADA, logowanie danych, sterowanie bezpośrednie) od producentów sprzętu

OPC w automatyce spełnia taką rolę jak w systemie operacyjnym sterownik drukarki

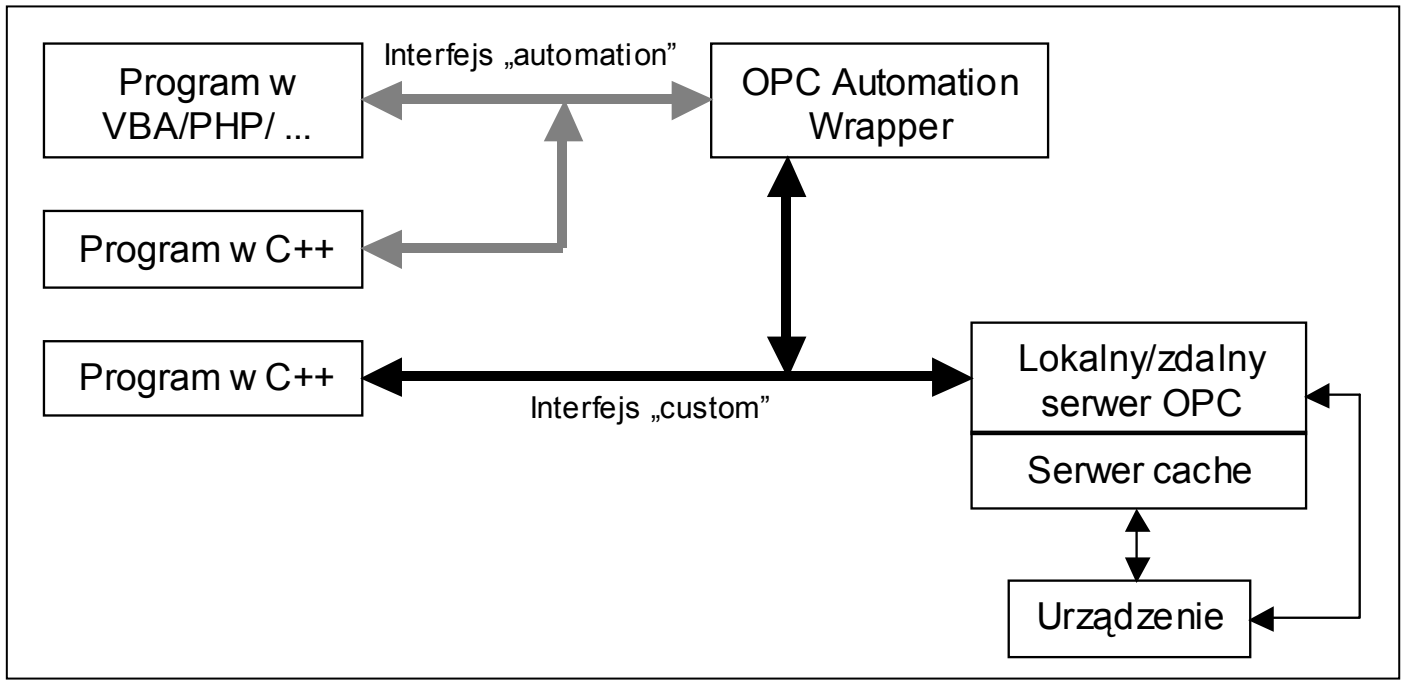
OPC



DCOM uzależnia OPC od Microsoft

OPC

- OPC wyróżnia dwa rodzaje interfejsów: custom oraz automation. Interfejsy typu custom są wykorzystywane w językach posiadających wskaźniki do funkcji (np.C). Języki nie posiadające wskaźników do funkcji (np. VB, PHP) wykorzystują interfejsy typu automation (tu metody nie są wywoływane przez wskaźniki do funkcji lecz przez ich nazwy). Automation Wrapper to DLL wykonujący odpowiednie konwersje.

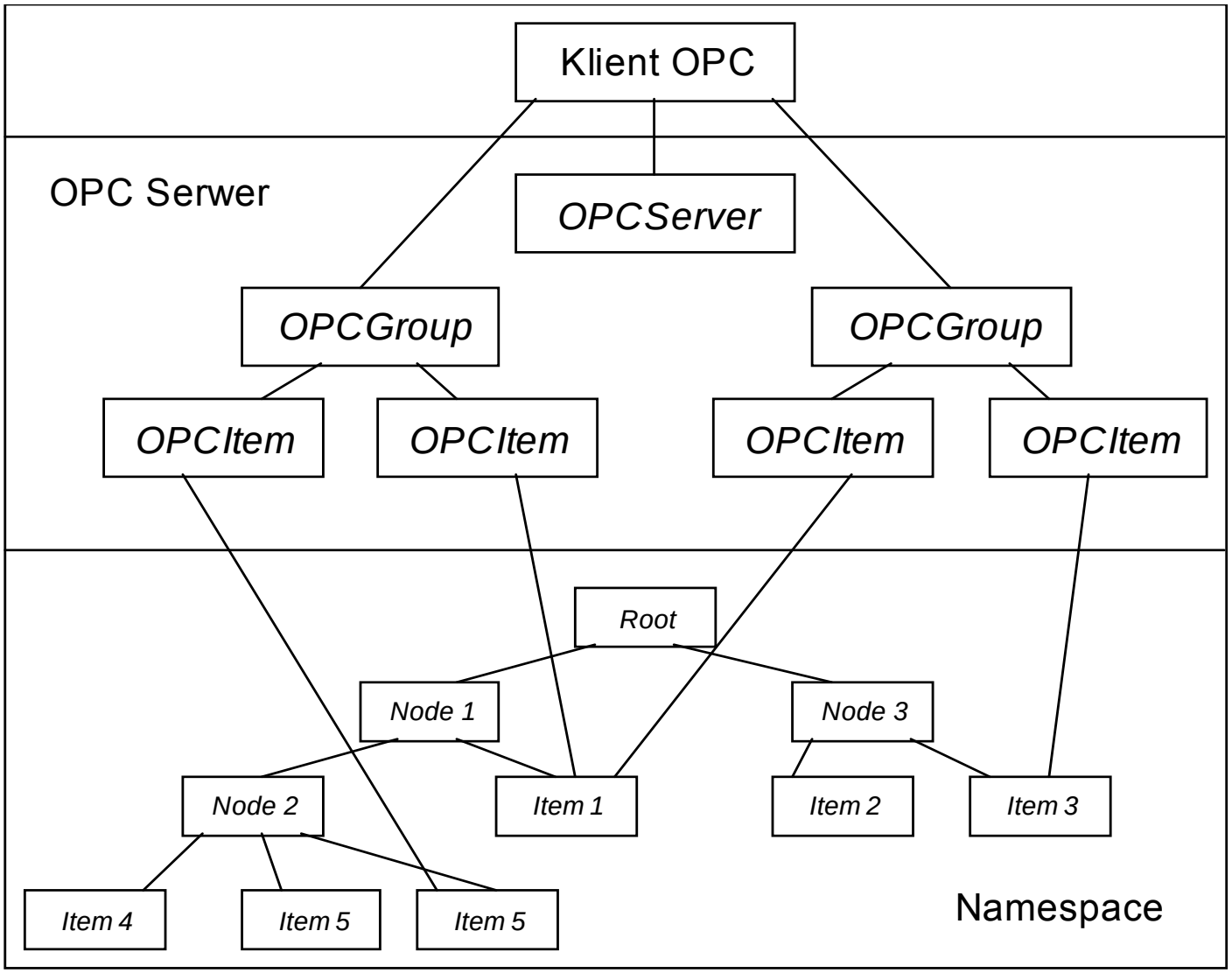


OPC Data Access Specification
(OPC Data Access Custom Interface Standard)

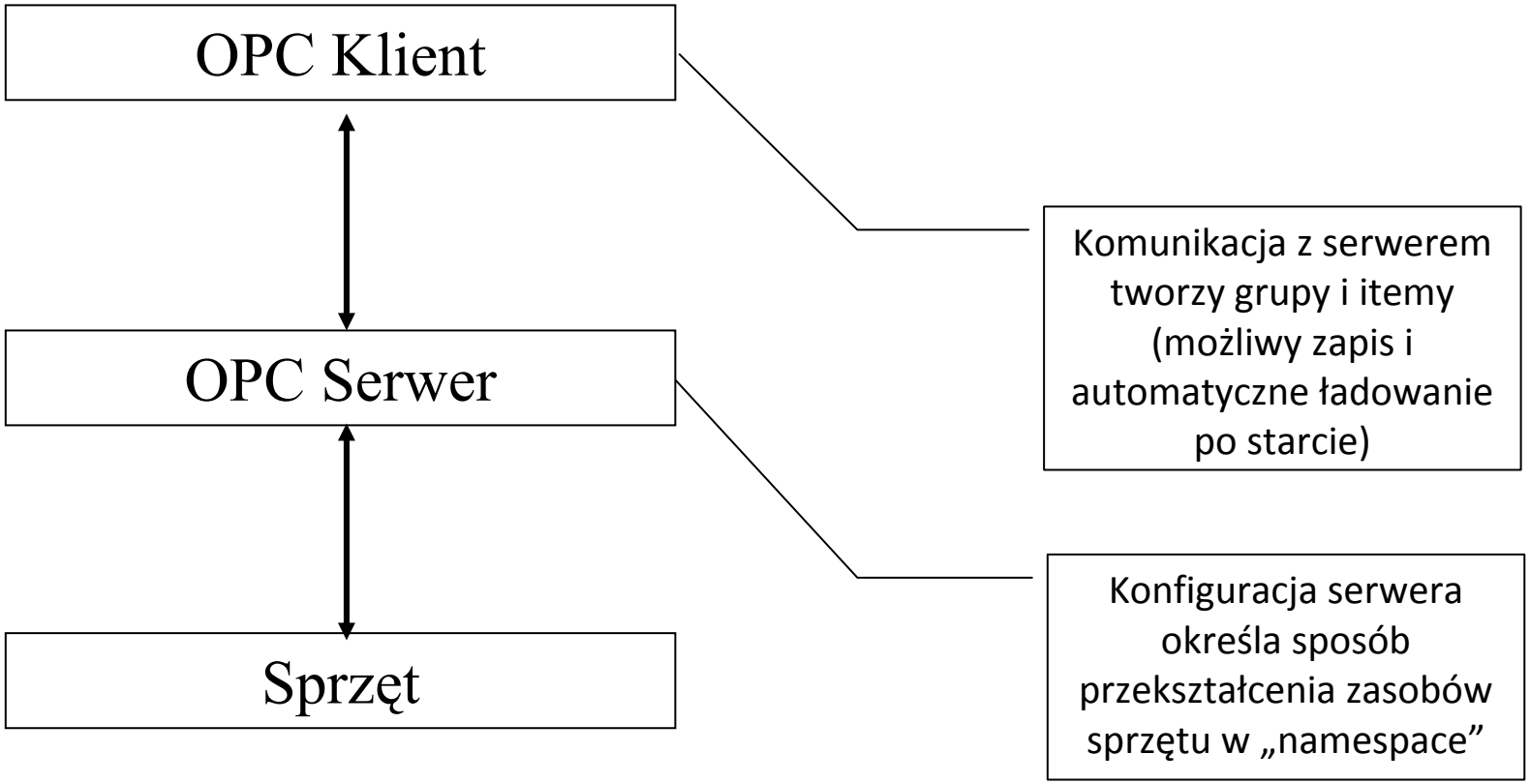
Dwa ważne pojecia:

- **przestrzeń nazw** (**namespace**) – wszystkie punkty produkujące i odbierające dane dostępne w serwerze. Może mieć strukturę płaską lub drzewa o dowolnej liczbie poziomów. Liście drzewa (producenci i konsumenci informacji zwane itemami) oraz węzły mogą posiadać dodatkowe właściwości (np. nazwę producenta czujnika). Z itemami związany jest również tzw. *Access Path* określający sposób komunikacji między urządzeniem a serwerem OPC (np. parametry transmisji po łączu szeregowym).
- **hierarchia obiektów OPC** (**OPC object hierarchy**) – klient OPC może tworzyć obiekty w serwerze OPC. To *OPCServer*, *OPCGroup* oraz *OPCItem*. Przestrzeń nazw występuje tylko jednokrotnie i jest wspólna dla wszystkich klientów OPC. Struktura obiektów jest indywidualna dla każdego klienta.

OPC


Klient OPC
OPC Serwer
OPCServer
OPCGroup
OPCGroup
OPCItem
OPCItem
OPCItem
OPCItem
Root
Node 1
Node 3
Node 2
Item 1
Item 2
Item 3
Item 4
Item 5
Item 5
Namespace

OPC


OPC Klient
OPC Serwer
Sprzęt
Komunikacja z serwerem tworzy grupy i itemy (możliwy zapis i automatyczne ładowanie po starcie)
Konfiguracja serwera określa sposób przekształcenia zasobów sprzętu w „namespace”

OPC

- W serwerze OPC tylko OPCServer oraz OPCGroup są rzeczywistymi obiektami DCOM. OPCItem nim nie jest – w rzeczywistości wykonuje się odczyt i zapis wszystkich wartości w grupie. Grupy strukturyzują itemy. Grupy tworzy się wewnątrz serwera.
- Utworzenie grupy wymaga przesłania do serwera:

  nazwy grupy,

  RequestedUpdateRate – częstotliwość skanowania aktywnych itemów

  PercentDeadband – strefa martwa dla trybu *Refresh*,

  ActiveState – gdy grupa nie jest aktywna zmienne procesowe nie są czytane automatycznie.
- Utworzenie itemu wymaga przesłania:

  Full qualified item ID – pełna nazwa,

  ActiveState

  RequestedDatatype – serwer wykona konwersję do wymaganego typu (w miarę możliwości)

  AccessPath – opcjonalny (np. dodatkowe parametry komunikacji z czujnikiem)

  ClientHandle – umożliwia rozróżnienie itemów o identycznych *item ID* tworzonych przez różnych klientów.
- Serwer OPC zawiera część niezależną od aplikacji (implementacja OPCServer oraz OPCGroup) i część zależną od aplikacji – OPCItem fizycznie komunikujące się ze sprzętem.

OPC

- Klient OPC może pobierać dane bezpośrednio z urządzenia (device) lub z wewnętrznej pamięci serwera (cache).
- Dane mogą być czytane synchronicznie lub asynchronicznie (Data Access Specification 2.0 dopuszcza asynchroniczny odczyt wyłącznie z urządzenia oraz nie dopuszcza asynchronicznego zapisu). W trakcie odczytu synchronicznego klient wywołuje metodę i czeka na zwrócenie wartości. Jest stosowany tylko gdy dostęp do danych jest szybki – w przeciwnym przypadku zablokuje klienta. W odczycie asynchronicznym po wywołaniu metody powrót jest natychmiastowy, a serwer po pewnym czasie powiadamia klienta o dostępności nowych danych (callback). Zapis zawsze wykonywany jest bezpośrednio do urządzenia (zapis do cache jest planowany w Data Access 3.0). Synchroniczny odczyt z cache jest możliwy gdy OPCGroup oraz OPCItem są aktywne.
- Synchroniczny lub asynchroniczny odczyt dużej liczby danych może być czasochłonny. W takim przypadku pracuje się w trybie *Refresh*. Serwer i klient wymieniają się tu rolami. Ten tryb pracuje tylko dla zmiennych analogowych posiadających informacje EU (Engineering Units – określa minimalną i maksymalną wartość sygnału). Wartość *PercentDeadband* wyznacza procentowy zakres wewnątrz EU po którego przekroczeniu (w stosunku do ostatnio wysłanej wartości) nowa wartość jest przesyłana od serwera do klienta. Serwer OPC powiadamia klienta tylko w przypadkach gdy konkretna wartość (lub jej status) uległa zmianie.

OPC

- Format przesyłanych danych miedzy serwerem a klientem zawiera:
  - aktualna wartość o typie VARIANT (char, short, int, long, boolean, float, double, Array, String)
  - znacznik czasu – 8-bajtowy licznik startujący od 1.01.1601 ze skokiem 100ns
  - status informacji – 2 bajty choć aktualnie tylko jeden używany. To 2 bity jakości danych (Good, Bad, Uncertain), 4 bity statusu (np. dokładne określenie co znaczy Bad) i 2 bity limitu (dodatkowa diagnostyka w przypadku awarii czujników)
- Oprócz wartości zmiennych procesowych dla każdego węzła i liścia przestrzeni nazewniczej są dostępne właściwości (np. numer seryjny lub położenie urządzenia dostarczającego daną). Zwracają one informacje o charakterze statycznym. Klient może przepytać węzeł lub liść i uzyskać listę właściwości, ich wartości, typy oraz opis. Niektóre właściwości są obowiązkowe np. data type oraz access rights. Posiadając pełna ścieżkę dostępu do własności można utworzyć OPCItem dla tej własności.
- Klient żąda od serwera danych w odpowiednim typie i serwer wykonuje (w miarę możliwości) odpowiednie konwersje. Dane przechowywane są jako VARIANT

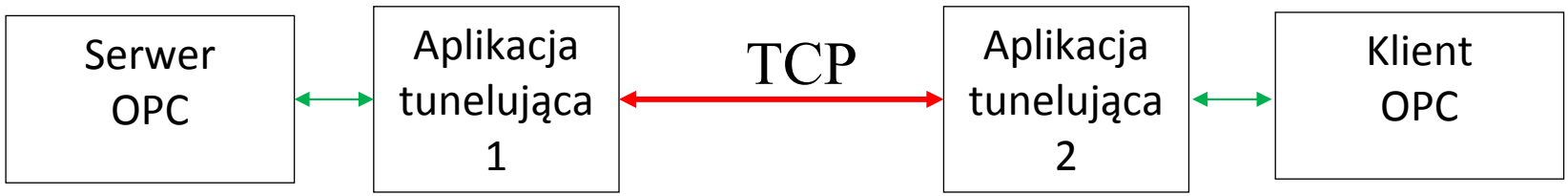
OPC

- Czasami korzystne jest aby *OPCGroup* oraz *OPCItem* pozostały po „zabiciu” klienta. Licznik referencyjny tych obiektów podczas tworzenia ustawiany jest na 2 – zabicie klienta dekrementuje ten licznik ale nie niszczy obiektów (NIEBEZPIECZNE !!!). Niszczeniu grup oraz itemów wymaga wywołania specjalnych metod. Pozostają one w serwerze mimo że nie ma aplikacji, które miała do nich uchwyty. Powtórne uzyskanie uchwytów możliwe jest dzięki wykorzystaniu enumeratorów (metod umożliwiających odszukanie) *OPCGoup* oraz *OPCServer*-ów.
- Utworzone grupy są prywatne dla tworzącego je klienta. Klient może upublicznić grupę i uwidocznić ją dla innych klientów. W stosunku do „swojej kopii” grupy publicznej klient może zmienić *UpdateRate*, *PercentDeadband* oraz *State*, a w stosunku do itemów *RequestedDatatype* oraz *State*. Nie może jednak dodawać oraz usuwać itemów.
- OPC Data Server może załadować z pliku konfigurację (istnieje do tego odpowiedni interfejs) z opisem namespace lub np. z parametrami konfiguracyjnymi.

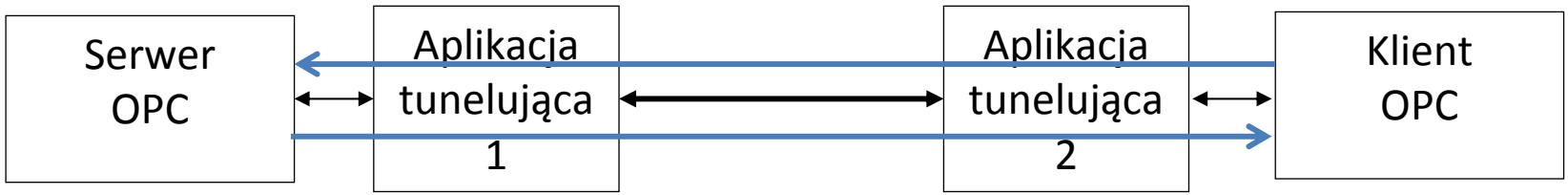
Tunelowanie

- Sieciowy ruch DCOM nie jest zaprojektowany dla czasu rzeczywistego
- DCOM – trudne do konfiguracji, źle reaguje na przerwanie połączenia i ma poważne braki dotyczące bezpieczeństwa
- Konfigurowanie OPC poprzez DCOM zwiększa ruch sieciowy

Rozwiązaniem opracowanie metod tunelowania polegających na zastąpieniu sieciowego ruchu DCOM poprzez TCP. Zamiast sieciowo łączyć klienta i serwer OPC łączy się je z lokalnymi aplikacjami. Lokalna aplikacja dla klienta udaje serwer, a lokalna aplikacja dla serwera udaje klienta. Lokalne aplikacje łączą się za pomocą TCP.

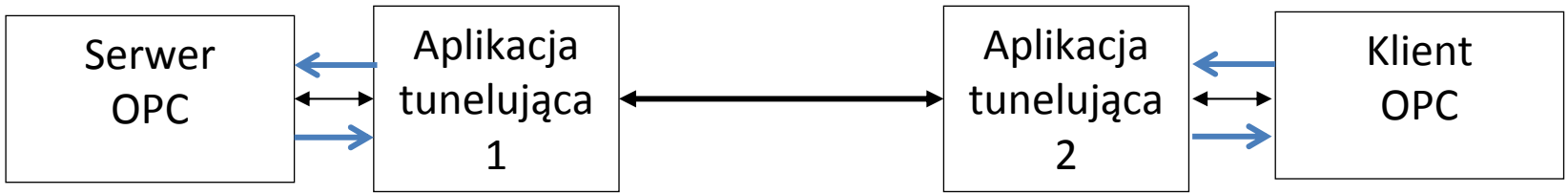


Tunelowanie transakcji OPC



- Transakcje OPC transmitowane są przez sieć
- Wywołania synchroniczne wrażliwe na parametry ruchu sieciowego

Lokalne transakcje OPC



- Transakcje OPC wykonywane wyłącznie lokalnie
- Aplikacje tunelujące utrzymują dwie kopie danych (serwera i klienta) w stanie permanentnej synchronizacji

- Aplikacje tunelujące mogą monitorować łącze i powiadamiać klienta/serwera o przerwaniu lub spadku jakości połączenia co może skutkować zmianą jakości danych

Bezpieczeństwo

- **Względy bezpieczeństwa najistotniejszym powodem wprowadzenia tunelowania**
- DCOM nie był projektowany do pracy w sieciach WAN, można go tak skonfigurować ale jest to trudne
- Aspekty sieciowego bezpieczeństwa:
  - Szyfrowanie danych – zapobiega podsłuchaniu danych
  - Uwierzytelnienie użytkowników – każde połączenie wymaga nazwy użytkownika i hasła (lub np. oparte na publicznym i prywatnym kluczu)
  - Autoryzacja – odpowiednie uprawnienia dla każdego użytkownika
- Rozwiązania wykorzystywane do zapewnienia bezpieczeństwa:
  - Opracować własne rozwiązanie od podstaw
  - SSL (Secure Socket Layer) – zapewnia wyłącznie szyfrowanie; bardzo wygodne; potrzebny klucz np. z centrum uwierzytelniania
  - VPN (Virtual Private Network) – zapewnia szyfrowanie i uwierzytelnianie; VPN jest częścią SO – tunelowanie odbywa się poprzez VPN
  - SSH (Secure Shell) - zapewnia szyfrowanie i uwierzytelnianie dla TCP; w odróżnieniu od VPN zabezpiecza tylko pojedyncze połączenie TCP

OPC Alarms and Events

- Mechanizm powiadamiania klienta AE o wystąpieniu zdarzenia lub alarmu (specyfikacja DA nie gwarantuje odczytania wszystkich wartości zmiennej)
- Alarm – warunek nienormalny; warunek to nazwany stan jednego z obiektów (np. HighTemperature)
- Zdarzenie – może ale nie musi być związane z jakimś warunkiem; to np. zakończenie stanu HighTemperature ale i działanie operatora
- Browser serwera AE uwidacznia wszystkie dostępne zdarzenia i alarmy
- Klient AE określa o jakich alarmach i zdarzeniach jest powiadamiany
- Klient może manipulować warunkami zaimplementowanymi na OPC AE serwerze
- W odróżnieniu od OPC DA nie dostarcza ciągłego strumienia danych
- Serwer OPC AE może być podłączony bezpośrednio do źródła danych lub do serwera OPC DA

OPC Historical Data Access

- Rodzaje serwerów HDA:
  - Simple trend – zapis strumieni danych (typowo z serwerów OPC DA) w postaci par [TimeStamp Value]
  - Complex – zapewniają kompesję i analizę danych; dostarczają danych w czasie jak i danych zagregowanych np. wartości minimalnej, średniej, itp.
- Dostępne są serwery HDA do relacyjnych baz danych lub do ODBC

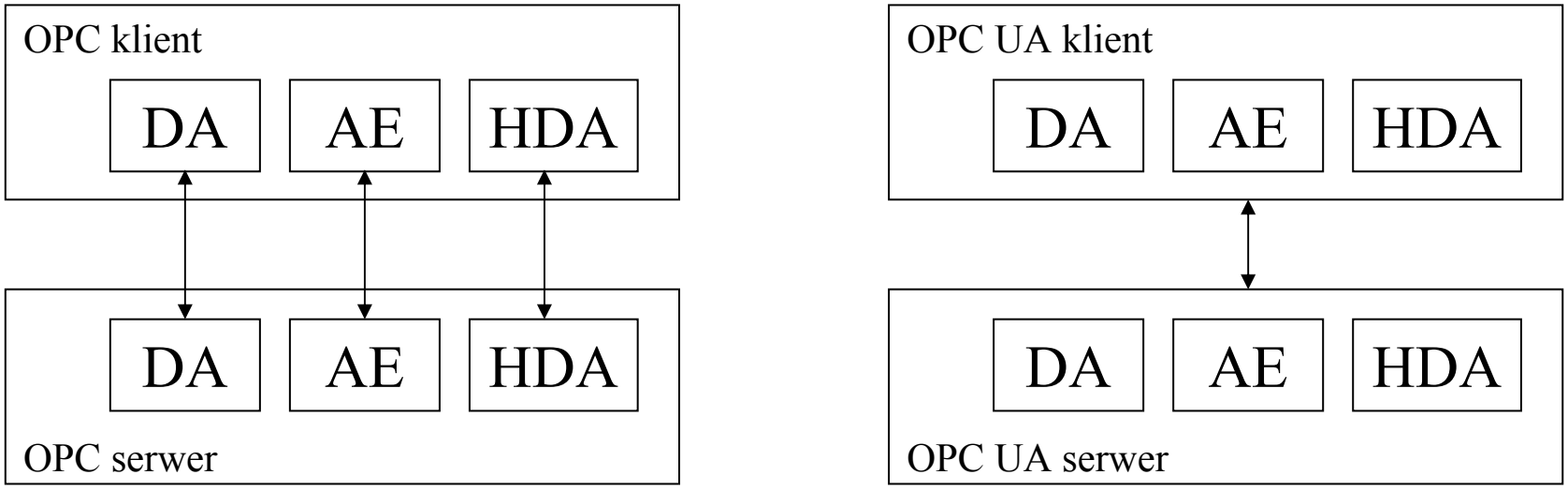
OPC Unified Architekture

- OPC UA – SOAP wykorzystany do zapewnienia funkcjonalności DA, EA oraz HDA

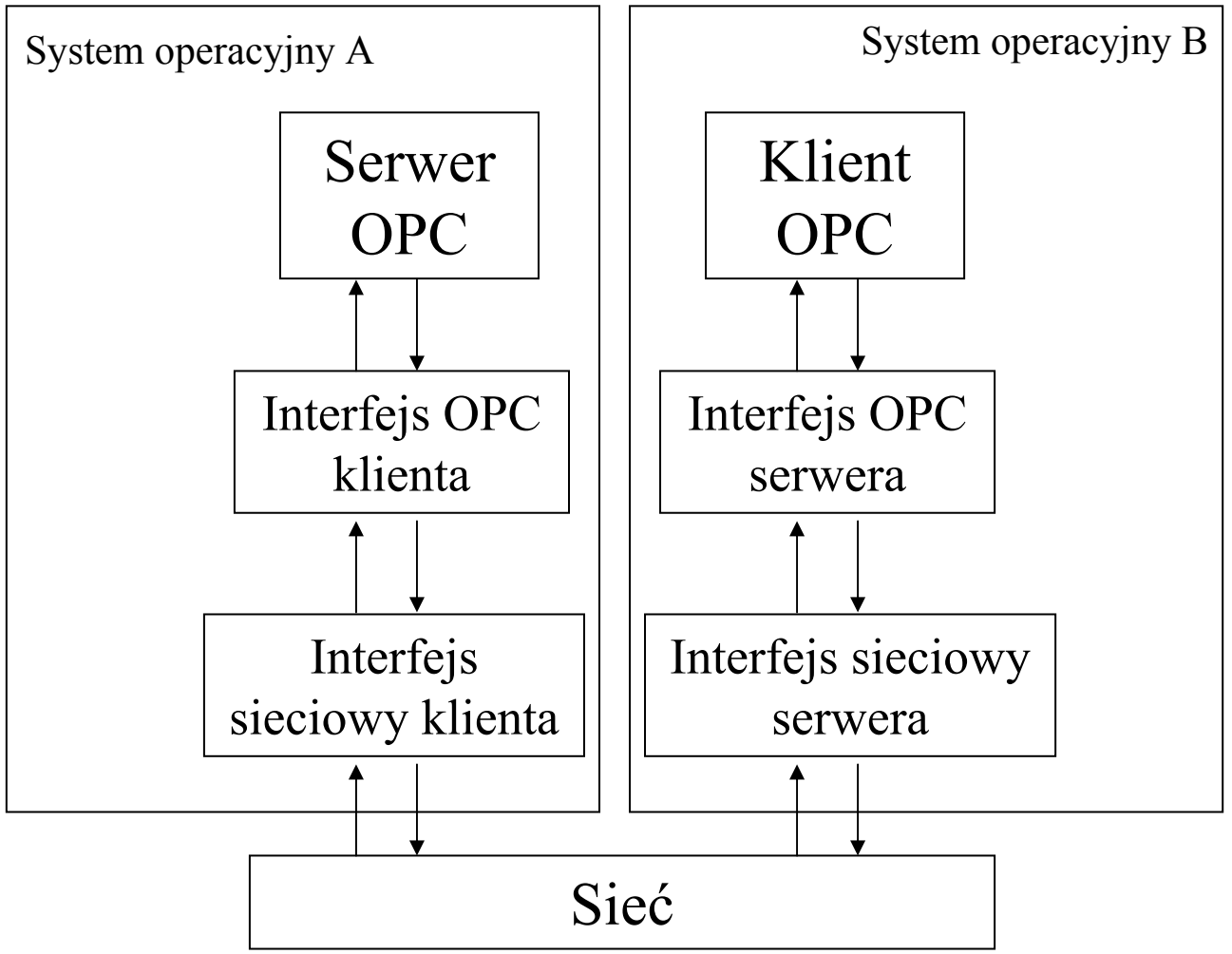
- Dwie specyfikacje
  - SOAP (Simple Object Access Protocol):
    - SOAP = XML + HTTP(S)
    - Protokół komunikacyjny wykorzystujący HTTP do przesyłania wiadomości XML
    - Niezależność od platformy i języka
  - Specyfikacja kodowania binarnego operująca bezpośrednio na komunikatach TCP; lepsza przepustowość od przesyłania wiadomości XML

- Klient może pytać serwer o metadane czyli np. o formaty w jakich udostępnia dane

OPC Unified Architekture



OPC Unified Architekture



Niezależność od platformy

Wykorzystanie OPC

	Zawsze/często	Czasem	Rzadko/nigdy
DA	82%	13%	5%
HDA	53%	15%	33%
A&E	42%	18%	40%
DX	25%	15%	60%
XML-DA	23%	14%	63%
Web Serwices	19%	12%	69%
Batch	17%	18%	65%
Security	15%	15%	70%
UA	10%	16%	74%

Wnioski:

Szeroki zakres wymagań czasowych oraz wymagań dotyczących oprogramowania wymaga stosowania różnorodnego sprzętu

Szeroki zakres typów kanałów we/wy

Specyficzne wymagania dotyczące systemów czasu rzeczywistego

Hierarchiczna struktura systemów sterowania – od sterowników PLC do wizualizacji za pomocą Internetu

Rozproszenie przetwarzania w układach sterowania wymaga stosowania sieci o deterministycznych charakterystykach

Układy rekonfigurowalne elastyczną alternatywą dla klasycznych modułów we/wy

Ciekawe możliwości wynikające z mieszania technologii np. układy FPGA oraz MATLAB

Liczne protokoły komunikacyjne wymiany danych między sprzętowymi i programowymi elementami automatyki

Literatura

1. Bill O. Gallmeister: *Programming for the real world. POSIX.4*, O'Reilly & Associates.
2. http://www.maximintegrated.com/app-notes/index.mvp/id/723
3. http://www.sterowniki.pl/index.php?option=Articles&task=viewarticle&artid=72
4. http://www.rtaautomation.com/ethercat/
5. http://lamspeople.epfl.ch/decotignie/RTN_intro.pdf
6. http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm
7. Norman Matloff: *Overview of Computer Networks*, http://heather.cs.ucdavis.edu/~matloff/Networks/Intro/NetIntro.pdf
8. *OPC Overview*, www.opcfoundation.org .
9. *Data Access Custom Interface Standard,* www.opcfoundation.org **.**