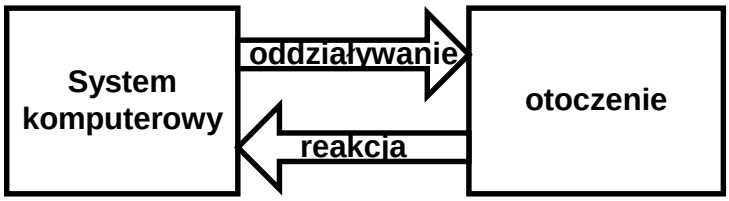
Systemy czasu rzeczywistego – wstęp

1 Systemy wbudowane

Często system sterujący stanowi integralną część urządzenia. Jest to system wbudowany. Nie posiada on żadnych elementów pozwalających na jego modyfikowanie (kompilator, edytor, konsola,...)



Rys. 1 Współdziałanie systemu komputerowego z otoczeniem

System wbudowany (*ang. embedded system*)

System wbudowany jest to system komputerowy będący częścią większego systemu i wykonujący istotną część jego funkcji. Przykładem może być komputer pokładowy samolotu lub system sterujący szybką koleją miejską.

System wbudowany to taki system który:

1. Jest częścią jakiegoś większego systemu,
2. Ma funkcjonować bez interwencji ze strony człowieka.

W systemach wbudowanych oprogramowanie tworzone jest na tak zwanym komputerze macierzystym (*ang. host*) a wykonywane na komputerze docelowym (*ang. target*).

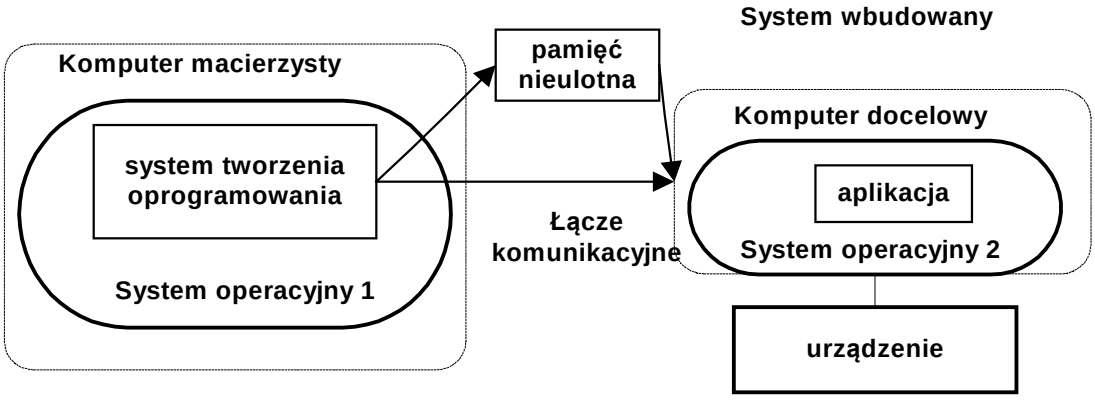
Tworzenie oprogramowania dla systemów wbudowanych

Na komputerze macierzystym pracującym pod kontrolą systemu operacyjnego 1 zainstalowany jest system tworzenia oprogramowania (*ang. development system*).

- edytor,
- kompilator,
- linker,
- biblioteki,
- debugger,
- symulatory,
- narzędzia do tworzenia dokumentacji

Utworzona w nim aplikacja wykonywana jest w komputerze docelowym (może być inny system operacyjny).

- Utworzona aplikacja wpisywana jest do pamięci komputera docelowego za pomocą interfejsu: JTAG, ISP, RS232
- wpisywana do pamięci nieulotnej (ROM, EPROM, Flash)



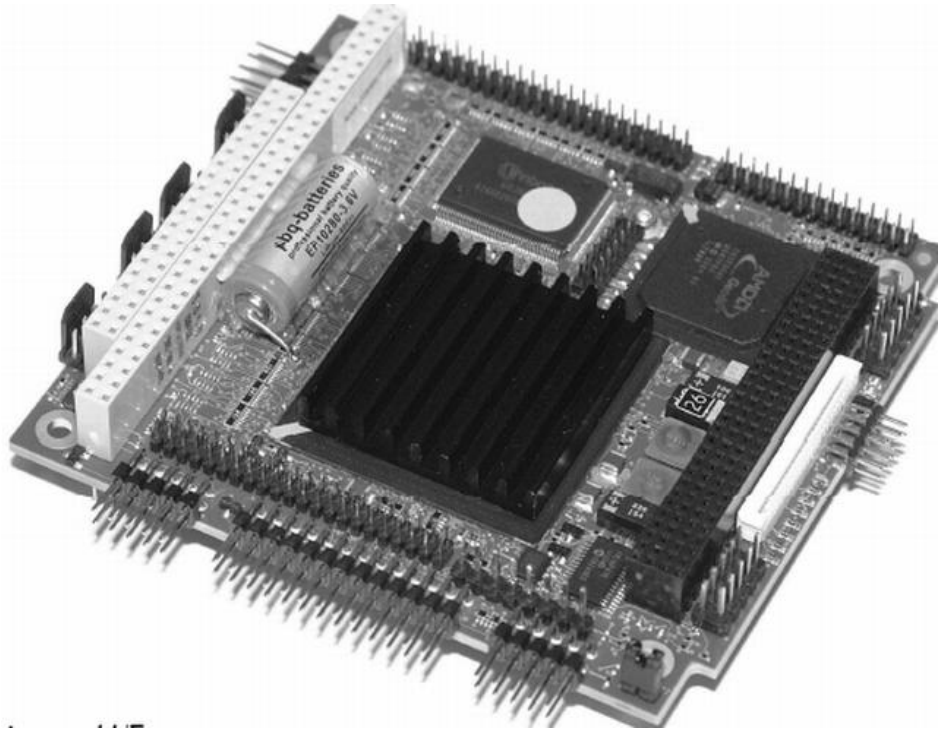
Rys. 2 **System skrośnego rozwoju oprogramowania**

Komputery przeznaczone do zastosowań wbudowanych posiadają pewne wspólne własności.

1. Wymagana jest odporność na pracę w trudnych warunkach otoczenia (wibracje, zapylenie, wilgoć), dopuszczalny jest szeroki zakres temperatur otoczenia.
2. Przeznaczone są do pracy ciągłej - brak jest elementów ruchomych (dyski obrotowe, wentylatory, napędy dyskietek), wymagana jest trwałość, łatwość serwisowania.
3. Oprogramowanie umieszczone jest w pamięci nieulotnej – ROM, flash, EPROM lub podobnej.
4. Stosowane jest wsparcie sprzętowe dla osiągania niezawodnej pracy –budzik (*ang. watchdog*), pamięci ECC, magistrala z kontrolą parzystości, poszerzona diagnostyka.

Standardy dotyczące komputerów przeznaczonych dla systemów sterujących i wbudowanych.

- VME,
- Compact PCI
- PC104



Rys. 3 Przykład komputera typu PC104 dla zastosowań wbudowanych MSM800SEV

2 Systemy czasu rzeczywistego

System czasu rzeczywistego (*ang. Real Time System – RTS*) jest systemem który współpracuje z zewnętrznym procesem. Musi on zapewniać wymagany czas reakcji na zewnętrzne zdarzenia.

System czasu rzeczywistego (*ang. Real-Time System*) jest to system komputerowy, w którym obliczenia prowadzone równolegle z przebiegiem zewnętrznego procesu maja na celu nadzorowanie, sterowanie i terminowe reagowanie na zachodzące w tym procesie zdarzenia.

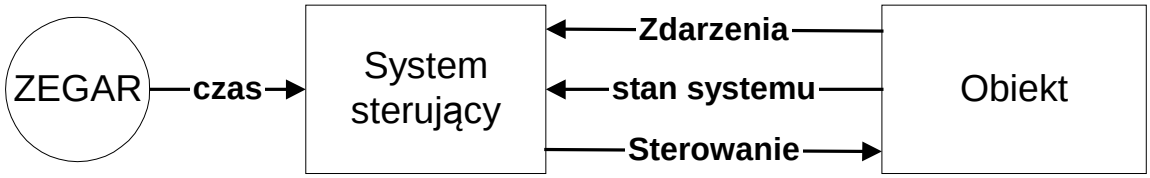
System czasu rzeczywistego jest takim systemem którego poprawność działania zależy od spełnienia warunków tak logicznych jak i czasowych.

1. Warunki logiczne – odpowiedź na zdarzenie przy uwzględnieniu stanu systemu musi być prawidłowa.
2. Warunki czasowe – odpowiedź musi nadejść we właściwym czasie.

Własności czasowe systemu RTS

System czasu rzeczywistego współdziała z otoczeniem. Jego zachowanie zależy od:

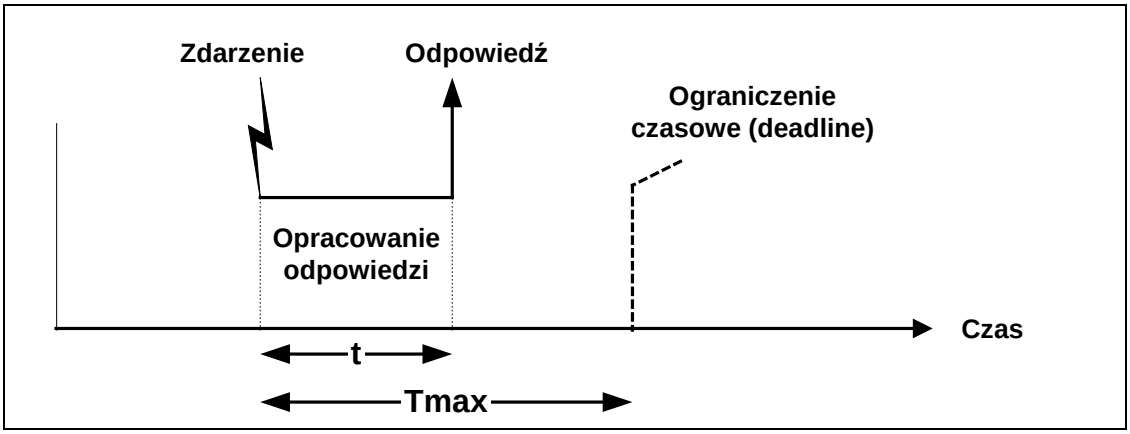
- stanu otoczenie
- czasu
- zdarzeń generowanych przez otoczenie



Rys. 4 Interakcje systemu z otoczeniem

System sterujący (*ang. Control system*)
System sterujący jest systemem komputerowym który ma utrzymywać nadzorowany obiekt w pożądanym stanie. Wymagana jest terminowa reakcja na zdarzenia generowane w nadzorowanym systemie.

Znaczenie ma nie tylko prawidłowość odpowiedzi na zdarzenie ale także czas tej odpowiedzi.



Rysunek 2-1 Definicja ograniczenia czasowego

System czasu rzeczywistego musi gwarantować że czas odpowiedzi t na zdarzenie musi być mniejszy od ograniczenia czasowego Tmax. (*ang. Deadline*)

Wartość wymaganego czasu odpowiedzi Tmax zależy od specyfikacji nadzorowanego systemu.

3 Rygorystyczne i łagodne systemy czasu rzeczywistego

Systemy RTS mogą znacznie różnić się od siebie pod względem konsekwencji niespełnienia ograniczeń czasowych.

Przykład:
- Odtwarzacz multimedialny
- Sterowanie silnikiem samolotowym

W związku z tym wyróżnia się kilka rodzajów systemów RTS.

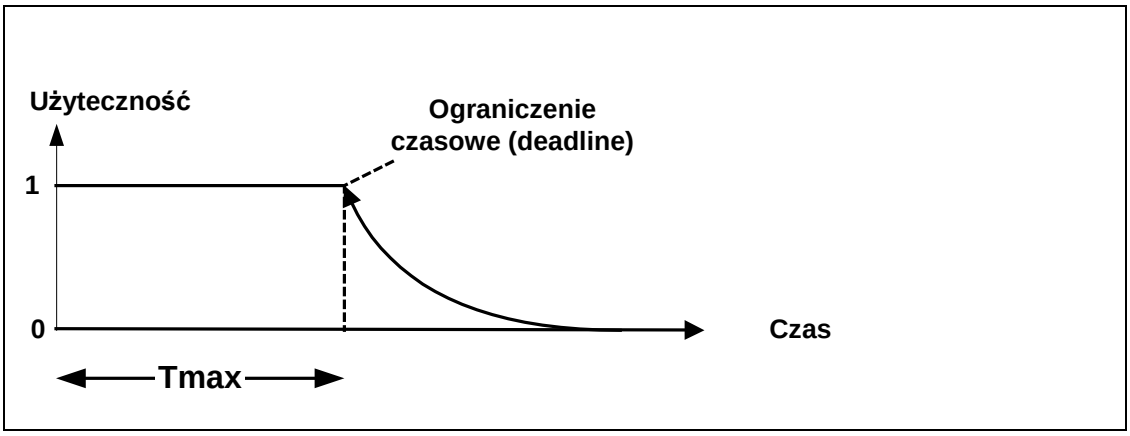
Rygorystyczne ograniczenie czasowe (*ang. Hard Deadline)* to takie ograniczenie które zawsze pozostaje spełnione. Jeśli choć raz zostało przekroczone uważa się że nie zostało spełnione.
Wymaga się aby istniała procedura walidacyjna pozwalająca na sprawdzenie czy warunek ten został spełniony.

Rygorystyczny system czasu rzeczywistego (*ang. Hard Real Time System*) to system w którym wymaga się spełnienia rygorystycznych ograniczeń czasowych.

Przykłady rygorystycznych systemów czasu rzeczywistego:
- System sterowania elektrownią atomową
- System sterowania samolotem
- System sterowania zapłonem samochodowym

Łagodne ograniczenie czasowe (ang. Soft Deadline) to takie ograniczenie czasowe które czasami może być przekroczone.

Jak zdefiniować pojęcie czasami:
1. Kategoria prawdopodobieństwa – np. ograniczenie spełnione jest w 99% przypadków.
2. Funkcja użyteczności – podaje ocenę korzyści w zależności od czasu uzyskania odpowiedzi.

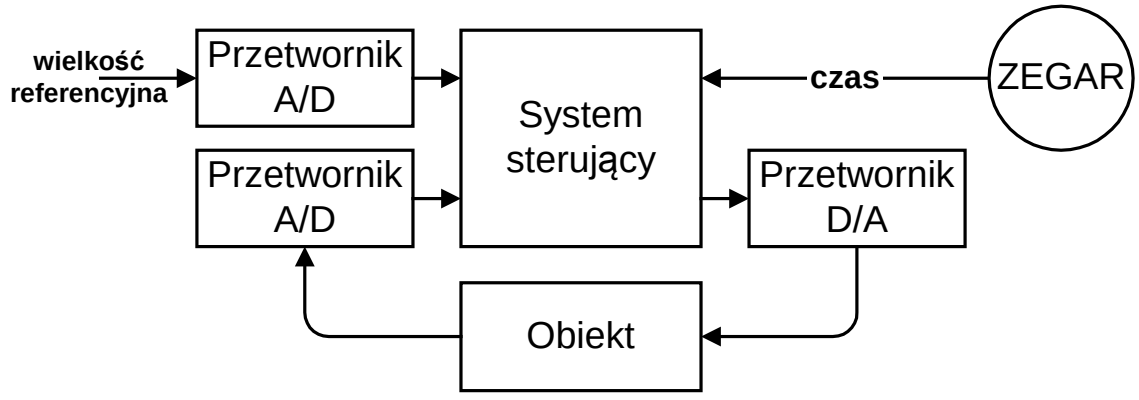


Rysunek 3-1 Funkcja użyteczności odpowiedzi

Łagodny system czasu rzeczywistego (*ang. Soft Real Time System*) to system w którym wymaga się spełnienia łagodnych ograniczeń czasowych.

Przykłady łagodnych systemów czasu rzeczywistego:

- Multimedia
- Sterowanie telefonem komórkowym
- Centrala telefoniczna



Rys. 5 Przykład systemu czasu rzeczywistego

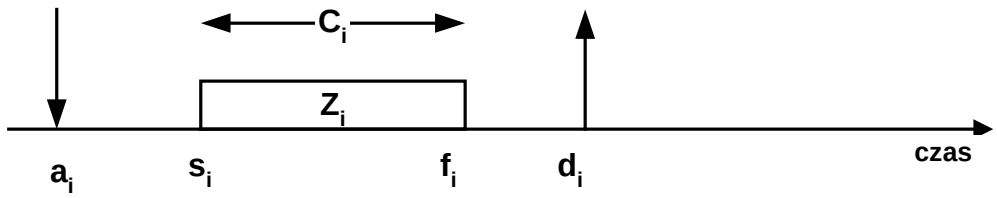
Charakterystyka zadań

W systemach czasu rzeczywistego wyróżniamy następujące typy zadań:

1. Zadania asynchroniczne (*ang. asynchronous*) – aktywowane przerwaniami.
2. Zadania synchroniczne (*ang. synchronous*) – aktywowane układami odmierzania czasu.
3. Zadania drugoplanowe (*ang. background*) – wykonywane w miarę wolnego czasu procesora.

Zadania asynchroniczne

Zadanie (asynchroniczne i synchroniczne) Z_i charakteryzuje się następującymi ograniczeniami czasowymi:



Rys. 6 Charakterystyki czasowe zadania

- Czas napłynięcia zadania (ang. *arrival time*) – a_i czas w którym pojawi się zadanie Z_i
- Czas wykonania zadania (ang. *computation time*) - c_i maksymalny czas potrzebny do wykonania zadania w sytuacji gdy wykonuje się na procesorze jako jedyne i ma ono wszystkie potrzebne zasoby.
- Ostateczny termin zakończenia (ang. *deadline time*) - d_i

4 Wymagania dla systemów RTS

1. Ciągłość działania – System RTS powinien pracować bez przerwy lub wtedy gdy się tego od niego wymaga

2. Zależność od otoczenia – System musi reagować na zdarzenia i dane powstające e zewnętrznym systemie.

3. Przewidywalność – Zdarzenia generowane przez otoczenie pojawiają się w przypadkowych momentach czasu. Może dojść do ich spiętrzenia. Mimo to system musi reagować zgodnie z wymaganiami (deterministycznie).

4.Terminowość – Reakcja na zdarzenia winna następować zgodnie z wymaganiami czasowymi tzn. nie może nastąpić zbyt późno.

Przewidywalność systemu ma miejsce wtedy gdy realizowane w systemie procedury niezależnie od okoliczności zajmują mniej więcej tyle samo czasu. Gdyby czasy wykonania tej samej procedury różniły się znacznie w zależności od obciążenia to przeszkadzałoby to w posługiwaniu się takim systemem i nie byłby to system przewidywalny.

Terminowość to zdolność do realizacji zadań których termin jest ściśle określony. Może to dotyczyć wykonania procedur które mają być reakcją na pojawiające się zdarzenia, wykonywania procedur cyklicznie z zadanym z góry interwałem, lub procedur dla których określony jest czas absolutny ich wykonania

W związku ze specyficznymi wymaganiami implementacja systemów RTS stanowi oddzielną klasę problemów.

5 Systemy operacyjne czasu rzeczywistego

W większości przypadków komputer sterujący systemem powinien pracować pod kontrolą systemu operacyjnego czasu rzeczywistego (*ang. Real Time Operating System - RTOS*).

Wymagania na system operacyjny czasu rzeczywistego:

1. Musi umożliwiać wykonywanie procesów wielowątkowych.
2. Wątki muszą posiadać priorytety.
3. Musi być stosowana wywłaszczająca strategia szeregowania.
4. Musi wspierać mechanizm przewidywalnej synchronizacji wątków.
5. Musi istnieć dziedziczenie priorytetów.
6. System musi być deterministyczny.
7. Opóźnienie obsługi przerwań powinno być znane i zależeć wyłącznie od ilości oczekujących przerwań;
8. Czas maskowania przerwań przez sterowniki i system operacyjny powinien być znany lub możliwy do przewidzenia.
9. Musi być pozbawiony błędów
10. Musi być dobrze udokumentowany.
11. Dobre wsparcie od dostawcy i stabilna pozycja na rynku

Aby zapewnić przenośność kodu źródłowego pomiędzy różnymi platformami wprowadza się normy na postać wywołań systemowych. Jedną z najważniejszych jest norma POSIX1003.4.

Norma POSIX 1003.4 wprowadza standard na rozszerzenia czasu rzeczywistego względem podstawowego standardu POSIX 1003.1. Rozszerzenia te obejmują funkcjonalności dane w poniższej tabeli.

Funkcja	Opis
Timery	Możliwość użycia czasomierzy wysokiej rozdzielczości
Szeregowanie wywłaszczające	Szeregowanie wywłaszczające uwzględniające priorytety procesów/wątków
Pamięć dzielona	Możliwość odwzorowania pamięci fizycznej w obszary dostępne dla procesów
Pliki czasu rzeczywistego	Możliwość użycia plików z deterministycznym czasem dostępu
Semafory	Możliwość użycia semaforów do synchronizacji procesów
Komunikacja międzyprocesowa	Możliwość użycia synchronicznych i asynchronicznych komunikatów
Zdarzenia	Obsługa zdarzeń asynchronicznych które mogą być kolejkowane i zachowują się deterministycznie
Blokowanie pamięci	Możliwość blokowania procesów w pamięci wirtualnej aby zapobiec ich przeniesieniu do pamięci zewnętrznej.
Asynchroniczny system wejścia/wyjścia	Możliwość nakładania się operacji przetwarzania i wejścia wyjścia

Tab. 5-1 Zestawienie funkcji czasu rzeczywistego normy POSIX1003.4

Nr	Nazwa	Producent	Platformy
1	Solaris	Sun Microsystems	Sparc
2	LynxOS	LynuxWorks	68K, MIPS, MPC8xx, PowerPC, x86, Sparc
3	VxWorks	Wind River Systems	68K, i869, ARM,MIPS, PowerPC, x86, SH, SPARC
4	QNX Neutrino	QNX Software Systems	MIPS, MPC8xx, PowerPC, SH, ARM, Strong RM
5	RT Linux	Open Source	ARM, PowerPC, x86, SH3, MIPS
6	Windows CE	Microsoft	ARM,MIPS,PowerPC, SH,x86, Strong ARM, NEC, VR4111
7	eCOS	Open Source	ARM,MIPS, MPC8xx, PowerPC, Sparc

Tab. 5-2 Przykłady systemów czasu rzeczywistego: