Własności systemu czasu rzeczywistego

System czasu rzeczywistego (według IEEE) musi wykonywać swe funkcje i odpowiadać na zewnętrzne wydarzenia asynchroniczne w ściśle przewidywalnym czasie.

Musi on mieć następujące własności:

1. priorytetowe przełączanie zadań z wywłaszczaniem;
2. możliwość bezpośredniego sterowania przełączaniem zadań przez użytkownika;
3. synchronizacja procesów w czasie rzeczywistym zapewniająca współużytkowanie zasobów systemu;
4. obsługa odmierzania czasu rzeczywistego;
5. obsługa przerwań sprzętowych zapewniająca wymagany czas reakcji.

Odmierzanie czasu w systemie OS–9

Funkcje modułu ***clock*** (jednorazowo, przy starcie systemu):

- inicjalizacja sprzętu generującego przerwanie cykliczne (*tick*);
- inicjalizacja zmiennych systemowych:
  - `D_TckSec` - ilość taktów na sekundę,
  - `D_Tick` - numer bieżącego taktu;
- instalacja procedury obsługi przerwania cyklicznego (zegarowego).

Funkcje procedury obsługi przerwania zegarowego:

- obsługa sprzętu generującego przerwanie cykliczne (*tick*);
- wywołanie systemowej procedury obsługi zegara w module ***kernel*** (adres w `D_Clock`).

Usługi związane z odmierzaniem czasu

F$Sleep – deaktywacja (uśpienie) procesu na zadaną ilość taktów zegara (*ticks*);

Parametry:
d0.l = zadana ilość taktów
Wynik:
d0.l = ilość pozostałych taktów przy za wczesnym obudzeniu

F$Alarm – utworzenie *timer*-a w celu wysłania sygnału do procesu po uplywie zadanego czasu.

Parametry:
d0.l = identyfikator alarmu (id)
d1.w = kod funkcji alarmu
d2.l = kod sygnału
d3.l = interwał lub czas alarmu
d0.l = data alarmu
Wynik:
d0.l = identyfikator alarmu (id)

Funkcje obsługi alarmów w OS–9

Nazwa	funkcja w C	Opis
A$Set	*alm_set(sc, tim)*	ustawienie alarmu względnego z sygnałem *sc* po czasie *tim*
A$Cycle	*alm_cycle(sc, tim)*	ustawienie alarmu cyklicznego *sc* z interwałem *tim*
A$AtDate	*alm_atdate(sc, tim, dat)*	ustawienie alarmu bezwzględnego *sc* w/g czasu gregoriańskiego (00HHMMSS, YYYYMMDD)
A$AtJul	*alm_atjul(sc, tim, dat)*	ustawienie alarmu bezwzględnego *sc* w/g czasu juliańskiego (sekundy od północy, dzień od roku -4712)
A$Delete	*alm_delete(id)*	usunięcie alarmu o numerze *id*

Struktura opisująca wątek alarmu (*Thread Execution Block*)

adr.	asm	znaczenie
00	T_ID	zarezerwowane
02	T_Proc	Identyfikator procesu–właściciela
04	T_MSiz	Wielkość pamięci wątku
08	T_User	Numer użytkownika
0C	T_Next	Wskaźnik na następny blok opisu wątku (lista uporządkowana według czasu budzenia)
10	T_Prev	Wskaźnik na poprzedni blok opisu wątku
14	T_Link	Wskaźniki listy wątków pokrewnych
1C	T_Cycle	Okres dla alarmu cyklicznego
20	T_WkTime	Czas obudzenia
24	T_WkDate	Data obudzenia
28		zarezerwowane
2C	T_Regs	Obraz rejestrów procesora

Obsługa funkcji czasowych

Przerwanie zegarowe budzi Proces Systemowy (ID = 1) o najwyższym z możliwych priorytecie (P$Prio = 65535).

Proces Systemowy:

1. Sprawdza kolejkę procesów śpiących (zagnieżdżoną w D_SleepQ), i budzi (F$AProc) odpowiednie procesy.

2. Obsługuje alarmy względne i cykliczne (lista zagnieżdżona w D_AlarTh):

   Jeśli T_WkTime $\leq$ D_Tick to:

   - wątek alarmu jednokrotnego jest uruchamiany i kasowany;

   - wątek alarmu cyklicznego jest uruchamiany i regenerowany: T_WkTime += T_Cycle.

3. Obsługuje alarmy bezwzględne (lista zagnieżdżona w D_Thread):

   Jeśli T_WkDate:T_WkTime $\leq$ D_Julian:D_Second to wątek jest uruchamiany i kasowany.

Przykład użycia alarmu cyklicznego

```
#include <stdio.h>
#include <errno.h>
#include <signal.h>

main(argc,argv)
int argc; char **argv;
{
  int a_ID, i=1;

  if(argc<2) return(-1);
  sscanf(argv[1],"%d",&i);
  if((a_ID = alm_cycle (SIGWAKE,(1<<31)|256*i))==-1)
    exit(_errmsg(errno," can’t install alarm\n"));

  printf(ąlmcycle, alarm_ID=0x%06x\n", a_ID);
  while(1)
  {
    sleep(0);
    system("date");
  };
}
```