www.asix.com.pl

Drajwer Bufor

Dok. Nr PLP6021
Wersja: 29-01-2010

ASKOM

ASKOM Sp. z o. o., ul. Józefa Sowińskiego 13, 44-121 Gliwice,
tel. +48 (0) 32 3018100, fax +48 (0) 32 3018101,
http://www.askom.com.pl, e-mail: office@askom.com.pl

1 Kanał Bufor

W systemie wizualizacji asix został zaimplementowany uniwersalny kanał transmisji o nazwie BUFOR, który pozwala na wymianę danych pomiędzy menedżerem komunikacyjnym Asmen, a dowolnym programem transmisji danych procesowych opracowanym przez użytkownika.

Kanał BUFOR jest realizowany przez dwa programy:

- uniwersalny drajwer (zwany dalej BUFOR, dostarczany przez ASKOM) zapewniający komunikację pomiędzy menedżerem komunikacyjnym Asmen i dowolnym programem użytkownika,
- drajwer użytkownika (zwany dalej USERDRV), zaimplementowany w postaci procesu działającego w środowisku WINDOWS 95/NT.

Wymiana danych pomiędzy przedmiotowymi programami jest realizowana poprzez plik wymiany (ang. memory mapped file). Synchronizacja dostępu do pliku wymiany jest realizowana przy wykorzystaniu obiektu typu mutex.

2 Opis zasobów drajwera użytkownika

Plik wymiany wykorzystywany do wymiany danych pomiędzy USERDRV i BUFOR musi być utworzony przez USERDRV. Struktury danych opisujących zmienne procesowe znajdujące się w tym pliku muszą być przygotowane przez USERDRV na etapie inicjacji drajwera i udostępnione poprzez deskryptor UserDesc, umieszczony na początku pliku wymiany. Struktura deskryptora UserDesc podana jest poniżej:

```
struct UserDesc
{
#define USER_NAME 15
byte name[USER_NAME];      // nazwa drajwera
word progMajor             // numer wersji USERDRV
word progMinor;            // porządkowy numer wersji USERDRV
word major;                // główny numer wersji protokołu
word minor;                // porządkowy numer wersji protokołu
word flags;                // charakterystyka sposobu pracy
word status;               // status drajwera
word items;                // liczba obsługiwanych zmiennych
                           // (liczba deskryptorów zmiennych w tablicy
                           // VarDesc)
DWORD varDescOffset;       // położenie tablicy deskryptorów zmiennych
                           // w  pliku wymiany (offset względem początku
                           // pliku wymiany)
HANDLE synchroMutex;       // uchwyt obiektu synchronizacji typu mutex
word globalRead;           // globalna flaga odczytu
word globalWrite;          // globalna flaga zapisu
word rezerwa1;
word rezerwa2;
};
```

Znaczenia poszczególnych pól struktury UserDesc podane są poniżej:

Name	- znaczenia informacyjne
progMajor	- znaczenia informacyjne
progMinor	- znaczenia informacyjne
major	- dla weryfikacji zgodności wersji BUFOR i USERDRV
minor	- dla weryfikacji zgodności wersji BUFOR i USERDRV
flags	- określa sposób pracy drajwera USERDRV. Pole ma postać flagi bitowej. Aktualnie zdefiniowane flagi to: BFLAG_USERTIME - USERDRV ustawia czasy odczytu zmiennych. BFLAG_AUTOREFRESH - USERDRV automatycznie odświeża dane bez oczekiwania na sygnał z BUFOR Pozostałe bity pola flags muszą być wyzerowane.
Status	- aktualny stan USERDRV. Sposób użycia pola nie jest jeszcze zdefiniowany. Początkowo powinien być wyzerowany.
globalRead	- flaga służąca do przekazywania zleceń odczytu z BUFOR do USERDRV. Szczegółowy opis użycia przy opisie wykonania operacji odczytu. Pole powinno być inicjowane na wartość 0,
globalWrite	- flaga służąca do przekazywania zleceń zapisu z BUFOR do USERDRV. Szczegółowy opis użycia przy opisie wykonania operacji zapisu. Pole powinno być inicjowane na wartość 0,
synchroMutex	- uchwyt obiektu synchronizacj typu mutex, synchronizującego dostęp BUFOR i USERDRV do wspólnego pliku wymiany.
rezerwa1	- pole rezerwowe, inicjowane na wartość 0,
rezerwa2	- pole rezerwowe, inicjowane na wartość 0.

Nazwa programu użytkownika, nazwa pliku wymiany oraz parametry przekazywane do programu użytkownika jako parametry linii komendy są deklarowane przy parametryzacji drajwera BUFOR (w sekcji Asmena w pliku 'ini').

3 Deskryptory zmiennych procesowych

Zmienne procesowe są opisane poprzez deskryptory, umieszczone w tablicy VarDesc. Tablica VarDesc umieszczona jest w pliku wymiany. Położenie tablicy względem początku pliku wymiany określa pole varDescOffset deskryptora UserDesc.

Wszystkie operacje realizowane przez BUFOR i USERDRV bazują na zawartości deskryptorów VarDesc.

Inicjowanie deskryptorów jest realizowane częściowo przez USERDRV na etapie uruchamiania procesu USERDRV (wpisanie typu, liczby elementów, offsetów buforów odczytu i zapisu), natomiast ich pełne zainicjowanie jest kończone przez BUFOR w fazie tworzenia bazy zmiennych procesowych Asmena.

Struktura deskryptora VarDesc podana jest poniżej:

typedef struct VarDesc VARD

```
struct VarDesc
{
word type;              // typ zmiennej
word items;             // maksymalna liczba elementów zmiennej (zmienna może
                        // być tablicą)
dword period;           // liczba sekund pomiędzy odświeżeniami wartości zmiennej
/* pola używane przy operacjach odczytu/odświeżania */
DWORD readBuffer;       // położenie początku bufora operacji odczytu/odświeżania
                        // względem początku pliku wymiany
word readDataStat;      // status odczytanej wartości zmiennej
word readQuery;         // status poleceń odczytu/odświeżania
word readResponse;      // status realizacji poleceń odczytu/odświeżania
dword readTime;         // czas operacji odczytu przekazywany Asmenowi
byte readRez1;          // pole rezerwowe
word readRez2;          // pole rezerwowe
/* pola używane przy operacjach zapisu */
DWORD writeBuffer;      // położenie początku bufora operacji odczytu/odświeżania
                        // względem początku pliku wymiany
word writeDataStat;     // status zapisu wartości zmiennej
word writeQuery;        // status poleceń zapisu
word writeResponse;     // status realizacji poleceń zapisu
byte writeRez1;         // pole rezerwowe
word writeRez2;         // pole rezerwowe
};
```

USERDRV inicjuje następujące pola struktury VARD:

Type	- używany przez BUFOR w celu sprawdzenia zgodności typu zmiennej oraz typu oczekiwanego przez funkcję przeliczającą zadeklarowaną dla tej zmiennej. Lista typów zmiennych zawarta jest w pliku nagłówkowym bufor.h,
Items	- maksymalna liczba elementów (dla zmiennych tablicowych),
ReadBuffer	- położenie bufora, służącego do odczytu zmiennych, względem początku pliku wymiany. Rozmiar bufora powinien być identyczny z rozmiarem zmiennej,
writeBuffer	- położenie bufora, służącego do zapisu zmiennych, względem początku pliku wymiany. Rozmiar bufora powinien być identyczny z rozmiarem zmiennej. Jeśli jest równy NULL, to zapis zmiennej jest zablokowany,
readRez1	- pole rezerwowe. Inicjowane na wartość 0,
readRez2	- pole rezerwowe. Inicjowane na wartość 0,
writeRez1	- pole rezerwowe. Inicjowane na wartość 0,
writeRez2	- pole rezerwowe. Inicjowane na wartość 0,

Pozostałe pola są ustawiane przez BUFOR w trakcie budowania listy zmiennych lub są polami synchronizującymi. W każdym przypadku, pola te powinny być początkowo wyzerowane.

Znaczenie pól inicjowanych prze BUFOR jest następujące:

Period	- okres odświeżania w sekundach. Ustawiany przez BUFOR. Ma znaczenie jedynie informacyjne. Asmen ze swojej strony liczy czasy odświeżeń i będzie

zlecał operacje odczytu w odpowiednich momentach. Drajwer USERDRV, który wykonuje automatyczne odświeżanie zmiennych może używać pola period do określenia częstości odczytów,

ReadTime - USERDRV przekazuje w polu readTime czas odczytu zmiennej w postaci sekund liczonych od 1.1.1980. Drajwer nie musi obsługiwać tego pola jeśli flaga BFLAG_USERTIME w deskryptorze UserDesc nie została ustawiona.

readDataStat
readQuery
readResponse - pola synchronizacji operacji odczytu. Znaczenie opisane w punkcie 6,
writeDataStat
writeQuery
writeResponse - pola synchronizacji operacji zapisu. Znaczenie opisane w punkcie 7,

4 Parametryzacja drajwera BUFOR

Parametryzacja drajwera BUFOR wymaga podania :

- nazwy pliku wymiany, który służy do wymiany danych pomiędzy drajwerem
- BUFOR i drajwerem USERDRV,
- nazwy programu, który zostanie załadowany przez drajwer BUFOR
- jako drajwer USERDRV,
- parametrów linii komendy, które należy przekazać do drajwera USERDRV.

Poniżej została podana deklaracja drajwera użytkownika, realizowanego przez program USER.EXE, wymieniającego dane z drajwerem BUFOR poprzez plik wymiany o nazwie PLIK_MMF. Do programu USER.EXE przekazywane są trzy parametry.

```
TEST = BUFOR, PLIK_MMF, USER, PAR1 PAR2 PAR3
```

Gdzie:

TEST - jest nazwą kanału logicznego, wykorzystującego drajwer użytkownika.

5 Deklarowanie zmiennych procesowych

Adres symboliczny zmiennej procesowej ma postać :

Iindex

gdzie:

index - indeks danej zmiennej w tablicy deskryptorów zmiennych drajwera (VarDesc). Pierwsza zmienna posiada indeks 1.

Pozostałe parametry w deklaracji zmiennej procesowej mają typowe znaczenie.

6 Uzyskiwanie dostępu do wspólnych danych

Ze względu na niezależną pracę BUFOR i USERDRV konieczne jest zapewnienie synchronizacji dostępu do wspólnych danych w trakcie realizacji wszelkich operacji związanych z odczytem lub zapisem zmiennych procesowych poprzez wspólny plik wymiany.

Synchronizacja przedmiotowego dostępu jest realizowana w oparciu o procedury API WIN32 operujące na obiekcie klasy mutex. Zakłada się, że mutex synchronizujący dostęp do wspólnego pliku wymiany będzie tworzony przez USERDRV na etapie jego inicjacji, a jego identyfikator będzie przekazany drajwerowi BUFOR poprzez pole synchroMutex deskryptora UserDesc.

Poniżej podano przykładowy kod źródłowy realizujący utworzenie mutexa:

```
HANDLE TestMutex;
struct UserDesc UserDesc;

short UtworzenieSemafora(void)
{
if (TestMutex = CreateMutex(NULL,   FALSE, NULL)  == NULL)
        {
        /* błąd utworzenia semafora. Powrót z błędem */
        return ERROR;
        }

/*  przekazanie identyfikatora drajwerowi BUFOR */
up->synchroMutex = TestMutex;
return OK;
}
```

Poniżej podano przykładowy kod realizujący synchronizowany dostęp do wspólnego pliku wymiany :

```
void ObsługaWspólnychDanych(void)
{
        /* oczekiwanie na dostęp do wspólnego pliku wymiany */
        WaitForSingleObject(TestMutex,INFINITE);
        /* bezpieczne działanie na wspólnych danych */

                .
                .
                .
        /* zwolnienie prawa dostępu do wspólnego pliku wymiany */
        ReleaseMutex(TestMutex);
}
```

7 Realizacja operacji odczytu

W operacjach odczytu zmiennej procesowej biorą udział pola readBuffer, readDataStat, readQuery, readResponse, readTime struktury VARD. Pola readTime, readDataStat i bufor readBuffer mogą być zmieniane tylko przez USERDRV. Pola readResponse i readQuery są zmieniane przez obie strony. Definicje wszystkich flag użytych w opisie oraz statusy wykonania operacji są zdefiniowane w pliku nagłówkowym bufor.h.

Poniższe operacje są wykonywane w przypadku drajwera USERDRV, który nie realizuje automatycznego odświeżania wartości zmiennych procesowych.

Sekwencja operacji dla drajwera BUFOR - inicjacja odczytu:

1/ uzyskanie prawa dostępu,

2/ jeżeli bit INREAD w polu readResponse (poprzedni odczyt jest ciągle wyzerowany) to zaniechanie inicjacji odczytu,

3/ wyzerowanie pola readResponse,

4/ ustawienie flagi REQUEST w polu readQuery,

5/ powtórzenie kroków 2/, 3/ i 4/ dla wszystkich zmiennych, których odczyt jest inicjowany,

6/ ustawienie pola globalRead w deskryptorze UserDesc na wartość READ_REQUEST,

7/ zwolnienie prawa dostępu.

Sekwencja operacji dla drajwera USERDRV - inicjacja odczytu na żądanie BUFOR :

1/ sprawdzenie pola globalRead w deskryptorze UserDesc. Jeżeli jest niezerowe, to należy je wyzerować, a następnie przejść do przeglądu wszystkich zmiennych według kroków 2/ ¸ 7/,

2/ uzyskanie prawa dostępu,

3/ sprawdzenie ustawienia flagi REQUEST w polu readQuery. Jeżeli ustawiony to wykonanie kroków 4/ i 5/,

4/ wyzerowanie pola readQuery,

5/ wewnętrzna inicjacja fizycznego odczytu i ustawienie flagi INREAD w polu readResponse,

6/ wykonanie kroków 3/, 4/ i 5/ dla wszystkich zmiennych,

7/ zwolnienie prawa dostępu.

Sekwencja operacji dla USERDRV - zakończenie odczytu:

1/ uzyskanie prawa dostępu,

2/ wpisanie nowej wartości do bufora readBuffer, ustawienie pola readDataStat oraz pola readTime (jeżeli drajwer USERDRV samodzielnie ustawia czas odczytu),

3/ ustawienie flagi DONE i wyzerowanie flagi INREAD w polu readResponse,

4/ powtórzenie kroków 2/ i 3/ dla wszystkich zmiennych, których odczyt został zakończony,

5/ zwolnienie prawa dostępu.

Sekwencja operacji dla BUFOR - zakończenie odczytu:

1/ uzyskanie prawa dostępu,

2/ dla wszystkich zmiennych, których flaga DONE w polu readResponse jest ustawiona, pobranie zawartości bufora readBuffer oraz pól readDataStat i readTime,

3/ zwolnienie prawa dostępu.

8 Realizacja operacji zapisu

W operacjach zapisu zmiennej procesowej biorą udział pola writeBuffer, writeDataStat , writeQuery, writeResponse. Pole writeDataStat może być zmieniane tylko przez drajwer USERDRV. Bufor writeBuffer jest zmieniany tylko przez drajwer BUFOR. Pola writeResponse i writeQuery są zmieniane przez obie strony. Definicje wszystkich flag użytych w opisie oraz statusy wykonania operacji są zdefiniowane w pliku nagłówkowym bufor.h.

Sekwencja operacji dla BUFOR - inicjacja zapisu:

1/ uzyskanie prawa dostępu,

2/ jeżeli bit INWRITE w polu writeResponse jest ustawiony (poprzedni zapis jest ciągle wykonywany), to zaniechanie inicjacji (status błędu lub ponowienie próby zapisu po pewnym czasie),

3/ wyzerowanie pola writeResponse,

4/ wpisanie do bufora writeBuffer nowej wartości, ustawienie flagi REQUEST w polu writeQuery,

5/ powtórzenie kroków 2/, 3/ i 4/ dla wszystkich zmiennych zapisywanych w danym cyklu pracy drajwera BUFOR,

6/ ustawienie pola globalWrite w deskryptorze UserDesc na wartość WRITE_REQUEST,

7/ zwolnienie prawa dostępu.

Sekwencja operacji dla USERDRV - inicjacja zapisu:

1/ sprawdzenie pola globalWrite w deskryptorze UserDesc. Jeżeli jest niezerowe, to należy je wyzerować, a następnie przejść do przeglądu wszystkich zmiennych w sposób opisany poniżej,

2/ uzyskanie prawa dostępu,

3/ jeżeli flaga REQUEST w polu writeQuery jest ustawiona, to wykonanie kroków 4/ i 5/,

4/ wyzerowanie pola writeQuery,

5/ wewnętrzna inicjacja fizycznego zapisu i ustawienie flagi INWRITE w polu writeResponse,

6/ powtórzenie kroków 3/, 4/ i 5/ dla wszystkich zmiennych procesowych,

7/ zwolnienie prawa dostępu.

Sekwencja operacji dla USERDRV - zakończenie zapisu:

1/ uzyskanie prawa dostępu,

2/ ustawienie pola writeDataStat,

3/ wyzerowanie flagi INWRITE i ustawienie flagi DONE w polu writeResponse,

4/ powtórzenie kroków 2/ i 3/ dla wszystkich zmiennych dla których zakończono operacje zapisu,

5/ zwolnienie prawa dostępu.

Sekwencja operacji dla BUFOR - zakończenie zapisu:

1/ uzyskanie prawa dostępu,

2/ jeżeli flaga DONE w polu writeResponse jest ustawiona, to pobranie statusu z pola writeDataStat,

3/ powtórzenie punktu 2/ dla wszystkich zmiennych, dla których zainicjowano zapis,

4/ zwolnienie prawa dostępu.

9 Plik nagłówkowy bufor.h

Definicje wszystkich flag użytych w opisie oraz statusy wykonania operacji są zdefiniowane w pliku nagłówkowym bufor.h. Poniżej podano zawartość tego pliku.

```
#define BFLAG_USERTIME      1
#define BFLAG_AUTOREFRESH 2

/* flaga żądania odczytu lub zapisu */
#define REQUEST             1

/* flagi opisujące stan realizacji operacji odczytu lub zapisu */
#define INREAD              1
#define INWRITE             1
#define DONE                2

/* typy zmiennych procesowych */

#define  BTYPE              1          // bajt
#define  ITYPE              2          // integer
#define  WTYPE              3          // unsigned integer
#define  LTYPE              4          // long
#define  DWTYPE             5          // unsigned long
#define  FPTYPE             6          // float type


/* statusy wykonania operacji zwracane przez drajwer */
#define  AVD_GOOD           0          // o.k.
#define  AVD_BAD            1          // dana nie istnieje
#define  AVD_FAIR           2          // dana z poprzedniego odczytu. Aktualny
                                       // odczyt zakończył się błędem

#define AVD_POOR            3          // dana sygnalizowana przez drajwer jako
                                       // mało wiarygodna
#define AVD_ERROR           4          // dana sygnalizowana przez drajwer jako zła
```

Spis treści