ПРИСТАП ДО СЕРИСКАТА ПОРТА И ПЕЧАТАЧОТ ОД BIOS-ОТ

Пристап до серискатa порта

Компјутерите од целиот свет комуницираат помеѓу себе и разменуваат податоци. Најголем дел од времето овие компјутери користат стандардна телефонска линија за оваа комуникација. Стандардната телефонска линија овозможува само поспор пренос на податоците, но затоа пак дозволува корисниците да комуницираат помеѓу себе од било кој дел на светот. Во сите случаи податоците се пренесуваат сериски, т.е. еден бит на информација во единица време.

Сериска картичка

Пренос на податоците е можен доколку во празен слот од PC компјутерот се вметне картичка за сериска комуникација. Овој тип на картичка овозможува пренос на податоци помеѓу два компјутери директно преку кабел или преку телефонски линии. Оваа комуникација може да се изврши само со помош на софтвер кој ќе ја контролира картичката (RS232, RS455, ..). BIOS-от го нуди овој софтвер со помош на 4 функции кои се повикуваат од прекинот 14h.

Пред да направиме дискусија за овие функции ќе го разгледаме протоколот за пренос на податоци.

Должина на збор: Како што покажува сликата, само две состојби на линијата (високо и ниско), 1 и 0 се дозволени. Линијата останува високо ако нема пренос на податоци. Ако состојбата на линијата се промени на ниско, примачот знае дека започнал пренос на податоци. BIOS функциите дозволуваат пренос на збор од 7 или 8 бита, зависно од должината на зборот. Ако линијата е ниско за време на преносот, тоа значи дека се пренесува бит со вредност 0. Најмалку значајниот бит од зборот се пренесува прв, додека најмногу значајниот бит се пренесува последен.

Паритет: Зборот што се пренесува може да биде проследен со паритет кој овозможува детекција на грешка за време на преносот. Паритетот може да биде непарен(even) или парен (odd). Тоа значи на пример за непарен паритет, ако зборот што се пренесува содржи 2 бита на високо тогаш паритетот ќе биде високо како целиот контингент што се пренесува содржи непарен број на високи состојби, или ако зборот што се пренесува содржи непарен број на битови на високо тогаш паритетот ќе биде ниско, како целиот контигент на битови кој се пренесува содржи непарен број на состојби на високо.

Стоп бит: Стоп бит сигналот сигнализира крај на преносот на податоци. Протоколот за пренос на податоци дозволува 1, 1.5 и 2 стоп бита.

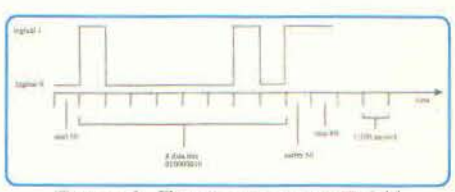
Брзина на пренос: Стандардот диктира брзина на пренос од 9600 baud (бита во секунда) и еден стоп бит. Стоп битот има обично вредност 1 и тоа значи дека линијата треба да се држи на високо за време од 1/9600 секунди, или 1/4800 секунди за 1.5 стоп бита.

Некои интерфејси работат со обратна логика, во тие случаи состојбата на 0 и 1 треба да се измени, што не го намалува принципот на објаснување.

Поставување на протокол: Преносот на податоци е можен кога и праќачот и примачот на податоците работат со еднакви параметри на пренос на податоците. Прво тоа е брзината на пренос. И предавателот и приемникот на податоците мора да биде поставен да работи на иста брзина на пренос што зависи од квалитетот на картичките, модемот и телефонската линија. Втор параметар кој треба да биде ист е должината на зборот која во случај на пренос на ASCII знаци може да биде 7 бита, но во случај на пренос на целото множество на знаци е 8 битна. Следен параметар е паритетот кој може да биде парен или непарен и служи за корекција на грешки при преносот. Последен параметар кој треба да се поклопува е стоп битот кој може да има 3 вредности. При вредност 1 се постигнува побрз пренос, додека при вредност 2 се постигнаува посигурен пренос.

Едноставен протокол:

На слика 1 е претставен едноставен пренос на знакот 'A' со ASCII код 65h=01000001b., со позитивна логика и брзина на пренос од 300 бауда и 1 стоп бит.



Слика 1. Пренос на знакот 'A'.

UART

Срцето на секоја сериска картица е UART (Universal Asynchronous Receiver Transmitter).

Регистри за испраќање: Знакот што се пренесува најпрво се пренесува до

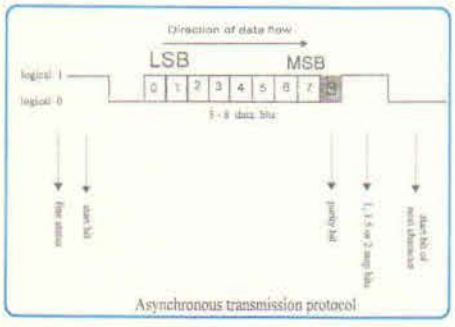
регистарот наречен transfer holding register. Потоа знакот се пренесува до transfer shift register од каде UART го пренесуба бит по бит по податочната линија, при тоа водејќи сметка за паритетот и стоп битот. Кога BIOS функцијата го пренесува статусот во AH регистарот, битовите 5 и 6 кога овие два регистра се празни.

Регистри за примање: Receiver shift register-от ги прима податоците (знакот) и потоа ги пренесува до receiver дата регистер од каде UART ги ослободува пренесениот податок од паритетот и стоп битот. Ако применият податок е сеуште во податочниот регистер, тогаш битот 1 од статусот на линијата се поставува на 1 за да се избегне презапишување со нов податок. Бит 0 покажува дека знакот е примен. Ако UART открие грешка на паритет при преносот тогаш го поставува битот 2 од статусот на 1. Битот 7 сигнализира дека е поминато времето за пренос. Ова се појавува кога комуникацијата не функционира нормално.

Bit 0 - Receive character
Bit 1 - Overwrite character in data register
Bit 2 - Parity error
Bit 3 - Protocol not specified
Bit 4 - Line interrupt
Bit 5 - Data register clear
Bit 6 - Shift register clear
Bit 7 - Time out

Функција 0: Поставување на протокол

Пред податоците да се пренесат или примат, UART мора да се информира за бројот на стоп битовите при преносот. Функцијата 0 од прекинот 14h го врши ова поставување. Бројот на функцијата се сместува во AH регистарот, а протоколот се сместува во AL регистарот. Битовите во AL регистарот имаат различно значење, како што е дадено во табелата:



Слика 2. Асинхрон протокол за пренос на податоци.

bit 2
Number of stop bits 0 - 1 stop bit
1 - 2 stop bit

bit 3, 4
Parity check 0 0(b) - none
01 (b) - odd
10 (b) - even

bit 5 - 7
Baud rate 000 - 110 Baud
001 - 150 Baud
010 - 300 Baud
001 - 600 Baud
100 - 1200 Baud
101 - 2400 Baud
110 - 4800 Baud
111 - 9600 Baud

Функција 1: Пренос на знак

Функцијата 1 пренесува знак. Пред повикувањето AH регистарот мора да содржи 1, а AL регистарот се полни со знакот кој треба да се пренесе. Ако знакот е пренесен тогаш битот 7 од AH регистарот се поставува на 0. Ако битот 7 од AH регистароте 1 тоа значи дека знакот не може да се пренесе.

Функција 2: Прием на знак

Функцијата 2 прима знак. По повикот на оваа функција AL регистарот го содржи применият знак. Ако е успешно извршен приемот тогаш AH има вредност 0.

Функција 3: Состојба на линија/модем

Функцијата 3 ја дава состојбата на линијата/состојбата на модемот. По извршување на оваа функција состојбата на линијата се враќа во AH регистарот а состојбата на модемот во AL регистарот. Во следната табела е дадена интерпретација на значењето на овие битови:

Bit 0 Modem ready to send status change
Bit 1 Modem on status change
Bit 2 Unused
Bit 3 Transfer error
Bit 4 1=Printer ON LINE, 0=Printer OFF LINE
Bit 5 Printer out of paper
Bit 6 Receive mode selected
Bit 7 Printer Busy

Пристап до печатачот од BIOS-от

BIOS-от нуди три функции, повикувани од прекинот 17h за комуникација со еден или повеќе печатачи прикачени на компјутерот. По секој од трите функциски повици, статусот на печатачот се сместува во AH регистарот. Секој бит во овој статусен бајт дава информации за статусот на работата на печатачот, дали има или нема хартија и.т.н.

Bit 0 Time out error

"Time out error" настанува кога компјутерот се обидува да испрати податоци а печатачот дава порака дека е зафатен (busy - битот 6 е 0) Бројот на обиди што BIOS-от ги прави за испраќање на податоци зависи од вредноста сместена на адресата 0040:0078 во RAM-от. ROM-от ја користи оваа адреса за сместување на променливи. Вредноста 20 што најчесто е сместена во оваа мемориска локација значи дека оваа бројка најпрво се множи со 4, а потоа со 65536, што дава приближно вредност од 5 милиони обиди. Имајќи предвид дека проверката се реализира со неколку асемблерски инструкции, тоа се случува релативно брзо.

Функција 0: Испраќање знак

Функцијата 0 испраќа знак до печатачот. Бројот на функцијата се сместува во AH регистарот а ASCII кодот на знакот што треба да се испрати во AL регистарот. По повикување на функцијата во AH регистарот се сместува статусниот бајт. Ако преносот не е успешно извршен, AH регистарот содржи вредност 1.

Функција 1: Иницијализирање на печатач

Оваа функција ги иницијализира портите на печатачот. Функцијата се извршува пред праќањето на податоци до печатачот. За нејзино активирање се сместува вредност 1 во AH регистарот (други параметри не се потребни).

Функција 2: Читање на стаусот на печатарот

Оваа функција го вчитува статусниот бајт во AH регистарот. Овој бајт како што е опишано погоре го дава статусот на печатачот. За нејзино активирање се сместува вредност 1 во AH регистарот (други параметри не се потребни).

Во продолжение е даден програмски пример во C за работа со портите на компјутерот со соодветни подпрограми. Целосниот листинг можете да го преземете од ИнФорма ЕИС од директориумот ИнФорма. E.O.T.

```
/*****************************************************************/
/*                          P L I N K C . C                      */
/*---------------------------------------------------------------*/
/*      Task          : Prenesuva datoteka preku paralelniot interfejs. */
/*---------------------------------------------------------------*/
/* Memoriski model  : SMALL                                      */
/*****************************************************************/
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <setjmp.h>
#include <string.h>
#ifdef __TURBOC__                              /* Turbo C compiler? */
  #include <dir.h>                        /* Include directory functions */
  #include <ctype.h>                            /* for toupper() */
#endif
/*== Type definitions ==========================================*/
typedef unsigned char BYTE;                          /* Definira byte */
typedef unsigned int WORD;                           /* Creira WORD */
typedef struct {                                /* Block transfer header */
                BYTE        Token;
                unsigned int Len;
               } BHEADER;
extern void IntrInstall( int far  * escape_flag,
                         WORD far * timeout_count );
extern void IntrRemove( void );
extern void EscapeDirect( int disconnect );
/*== Constanti =================================================*/
#define ONESEC        18                                /* Edna sekunda */
#define TENSEC        182                              /* Deset secundi */
#define TO_DEFAULT    TENSEC                 /* Time out tek. vrednost */
#define TRUE          ( 0 == 0 )
#define FALSE         ( 0 == 1 )
#define MAXBLOCK      4096          /* 4K maksimalna golemina na blok */
#define ACK           0x00                              /* Potvrdeno*/
#define NAK           0xFF                          /* Ne e potvrdeno */
#define MAX_TRY       5                      /* Maksimalen broj na probi */
#define TOK_DATSTART  0                    /* Pocetok na podat. na datot. */
#define TOK_DATNEXT   1                      /* Sleden blok na podatoci*/
#define TOK_DATEND    2                   /* Kraj na prenosot na podatoci */
#define TOK_ENDIT     3                        /* Kraj na programot */
#define TOK_ESCAPE    4 /* <Esc> pritisnato na dalecniot kompjuter */
/*-- Codes for LongJump call ---------------------------------*/
#define LJ_OKSENDER   1       /* Sits podatoci se prenesani uspesno */
#define LJ_OKRECD     2        /* Site podatoci se primeni uspesno */
#define LJ_TIMEOUT    3           /* Time out: */
#define LJ_ESCAPE     4          /* <Esc> pritisnato na lok. komp. */
#define LJ_REMESCAPE  5     /* <Esc> pritisnato na dalecniot kompj. */
#define LJ_DATA       6                 /* Greska vo komunikacijata */
#define LJ_NOLINK     7                          /* Nema vrska */
#define LJ_NOPAR      8                       /* Nema interfejs */
#define LJ_PARA       9            /* Nedobri povikuvacki parametri*/
/*== Macro =====================================================*/
#ifdef __TURBOC__                          /* Turbo C compiler? */
  #define GetB()                        ( inportb( InpPort ) & 0xFF )
  #define PutB( Was )                   outportb( OutPort, Was )
  #define DIRSTRUCT                     struct ffblk
  #define FINDFIRST( path, buf, attr )  findfirst( path, buf, attr )
  #define FINDNEXT( buf )               findnext( buf )
  #define DFILENAME                     ff_name
#else                                       /* No -> Microsoft C */
  #define GetB()                        ( inp( InpPort ) & 0xFF )
  #define PutB( Was )                   outp( OutPort, Was )
  #define DIRSTRUCT                     struct        find_t
  #define FINDFIRST( path, buf, attr )  _dos_findfirst(path, attr, buf)
  #define FINDNEXT( buf )               _dos_findnext( buf )
  #define DFILENAME                     name
#endif

#ifdef MK_FP                          /* Macro MK_FP already defined? */
  #undef MK_FP                             /* Yes -> Undefine macro */
#endif

#define MK_FP(seg,ofs) ((void far *) ((unsigned long) (seg)<<16|(ofs)))

/*== Global variables ==========================================*/

int     InpPort;                                 /* Vlezna port adresa */
int     OutPort;                                 /* Izlezna port adresa */
int     Escape = 0;                         /* <Esc> ne e pritisnato */
WORD    Timeout = TO_DEFAULT;
WORD    TO_Count;                              /* Brojac za  time out */
jmp_buf ReturnToEnd;                       /* Povratna adresa za kraj */
BYTE    *BlockBuf;
FILE    *FiVar = NULL;

/*****************************************************************/
/* GetPortAdr: Initializira paralelni port adresi vo             */
/*             globalnite promenlivi INPPORT  i  OUTPORT.        */
/* Input: LPTNUM = Parallel interface number (1-4)               */
/* Output: TRUE if interface is valid                            */
/* Global vars.: InpPort/W, OutPort/W                            */
/*****************************************************************/

int GetPortAdr( int LptNum )
{                   /* Gi cita adresite na portata od BIOS segmentot */
 OutPort = *( WORD far * ) MK_FP( 0x0040, 6 + LptNum * 2 );
 if ( OutPort != 0 )                          /* Interface dostapen */
  {                                                        /* Da */
   InpPort = OutPort + 1;              /* Vlez na adresa na registarot */
   return TRUE;
  }
 else
  return FALSE;                     /* Greska: Interfejsot ne e dostapen */
}

/*****************************************************************/
/* Port_Init      : Initializira registri potrebni za prenos     */
/* Input          : SENDER = TRUE if sender, FALSE if receiver   */
/* Output         :,TRUE if register initializes successfully    */
/* Global vars.   : InpPort/R, OutPort/R                         */
/*****************************************************************/

int Port_Init( int Sender )
{
 EscapeDirect( TRUE );
 if ( Sender )
  {
   TO_Count = Timeout * 5;                 /* Start an time out brojacot */
   PutB( 0x10 );                                    /* Praka: 00010000b */
   while ( ( GetB() != 0x00 ) && TO_Count )         /* Ceka na 00000000b */
    ;
  }
 else
  {
   TO_Count = Timeout * 5;                /* Pocetok na time out brojacot */
   while ( ( GetB() != 0x00 ) && TO_Count )      /* Ceka na  00000000b */
    ;
   PutB( 0x10 );                                  /* Ispraka: 00010000b */
  }
 EscapeDirect( FALSE );
 return ( TO_Count != 0 );                     /* Kraj na inicijalizacija */
}

/*****************************************************************/
/* SendAByte      : Ispraka byte do dalecniot kompjuter vo dva dela  */
/*                  potoa go ispituva resultatot.                    */
/* Input          : B2Send = Byte to be sent                         */
/* Output         : Transfer successful? (0 = error, -1 = O.K.)      */
/* Global vars.   : Timeout/R, InpPort/R, OutPort/R (in macros)      */
/*****************************************************************/

int SendAByte( BYTE B2Send )
{
 BYTE RcvdB;                                    /* Prifateniot byte */
TO_Count = Timeout;              /* Go inicijalizira  time out brojacot */
 PutB( B2Send & 0x0F );                          /* Ispraka, brisi  BUSY */
 while ( ( ( GetB() & 128 ) == 0 ) && TO_Count )   /* Ceka na poraka */
  ;
 if ( TO_Count == 0 )                             /* Time out greska? */
  longjmp( ReturnToEnd, LJ_TIMEOUT );           /* Prekin na prenosot */
 RcvdB = ( GetB() >> 3 ) & 0x0F;                  /* Bits 3-6 in 0-3 */
TO_Count = Timeout;              /* Go inicijalizira  time out brojacot */
 PutB( B2Send >> 4 ) | 0x10 );                    /* Ispraka BUSY */
 while ( ( ( GetB() & 128 ) != 0 ) && TO_Count )   /* Ceka na poraka */
  ;

 if ( TO_Count == 0 )                             /* Time out greska? */
  longjmp( ReturnToEnd, LJ_TIMEOUT );          /* Go prekinuva prenosot */

 RcvdB = RcvdB | ( ( GetB() << 1 ) & 0xF0 );      /* Bits 3-6 in 4-7 */
 return ( B2Send == RcvdB );                  /* Byte-ot e ispraten korektno */
}

/*****************************************************************/
/* ReceiveAByte : Go prifaka byte-ot od dalecniot kompjuter  i   */
/*                go fraka delot za testitanje.                  */
/* Input          : None                                         */
/* Output         : Received byte                                */
/* Global vars.   : Timeout/R, InpPort/R, OutPort/R (in macros)  */
/*****************************************************************/

BYTE ReceiveAByte( void )
{
 BYTE LoNib, HiNib;                             /* Prifaka nibbles */
TO_Count = Timeout;              /* Go inicijalizira time out brojacot */
 while ( ( ( GetB() & 128 ) == 0 ) && TO_Count ) /* Ceka dodeka BUSY 1 */
  ;

 if ( TO_Count == 0 )                             /* Time out greska? */
  longjmp( ReturnToEnd, LJ_TIMEOUT );          /* Go prekinuva prenosot */

 LoNib = ( GetB() >> 3 ) & 0x0F;                  /* Bits 3-6 in 0-3 */
 PutB( LoNib );                                   /* Povtorno praka */
 TO_Count = Timeout;              /* Go inicijalizira time out brojacot */
 while ( ( ( GetB() & 128 ) != 0 ) && TO_Count ) /* Ceka dodeka BUSY 0 */
  ;

 if ( TO_Count == 0 )                             /* Time out greska? */
  longjmp( ReturnToEnd, LJ_TIMEOUT );          /* Go prekinuva prenosot */

 HiNib = ( GetB() << 1 ) & 0xF0;                  /* Bits 3-6 in 4-7 */
 PutB( ( HiNib >> 4 ) | 0x10 );       /* Povtorno praka, postavuva BUSY */

 return( LoNib | HiNib );                                /* Primen byte */
}

/*****************************************************************/
/* SendABlock: Ispraka podatocen blok                            */
/* Input      : TOKEN  = Token for receiver                      */
/*              TRANUM = Number of bytes to be transferred       */
/*              DPTR   = Pointer to buffer containing data       */
/* Output     : None, jumps immediately to an error handler if   */
/*              an error occurs.                                 */
/*****************************************************************/

void SendABlock( BYTE Token, int TraNum, void *DPtr )
{
 BHEADER header;
 BYTE    *bptr,     /* Pokazuvac na tekovnoit bajt sto treba da se isprati */
         RcvrEscape;                  /* <Esc> pritisnat na dalecn. kompj.? */
 int     ok,                                              /* Greska flag */
         i,                                  /* brojac na povtoruvanja */
         try;                               /* Predstanat broj na probi */

 if ( Escape )                    /* <Esc> pritisnato na lokalniot kompjuter?*/
  {
   Token = TOK_ESCAPE;                                          /* Da? */
   TraNum = 0;
  }
header.Token = Token;             /* Kreira header */
 header.Len = TraNum;

 for ( try = MAX_TRY; try; --try )
  {
   ok = TRUE;                                  /*     PRODOLZHUVA      */
```