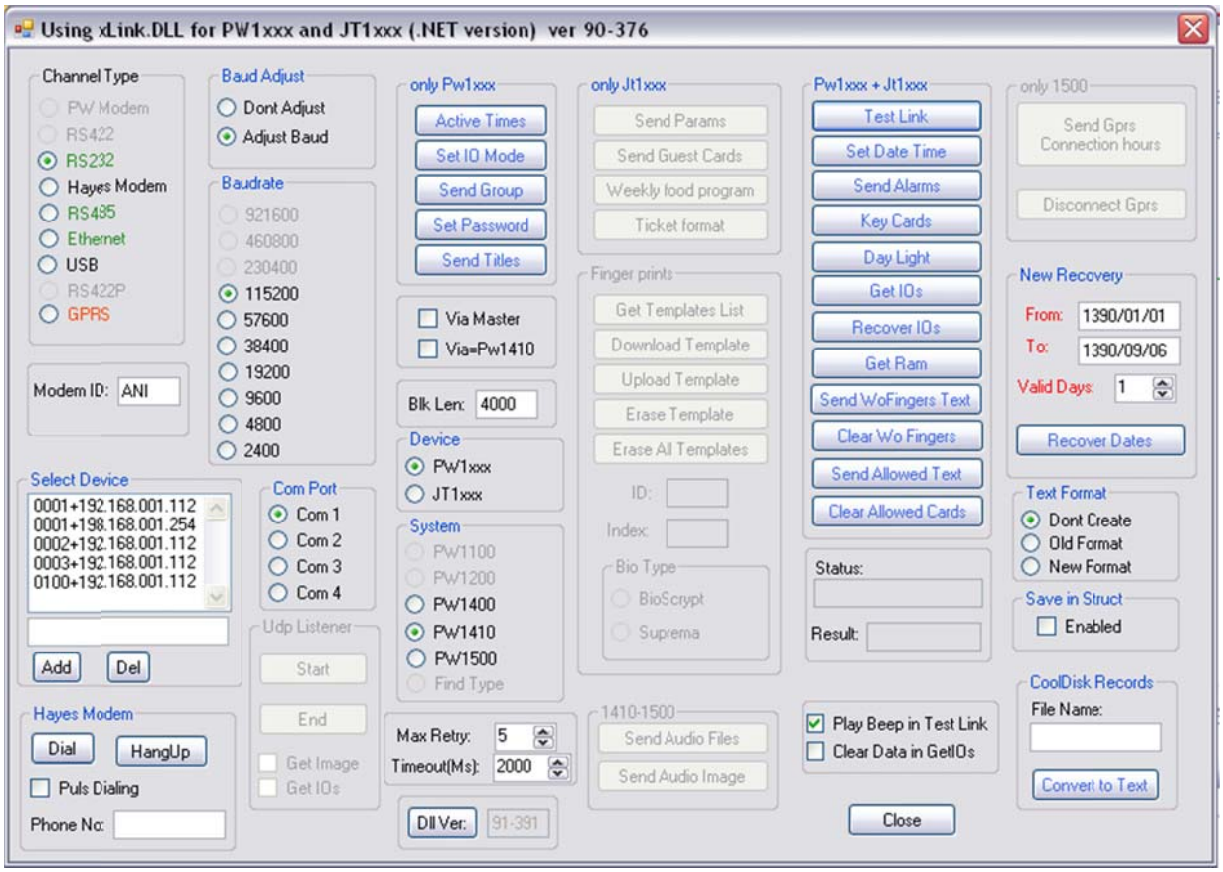
راهنمای برنامه ارتباطی Sample

برای کار با برنامه sample منوی فرم بصورت زیر مشاهده خواهد شد:



در حال حاضر دستورات آبی رنگ در این فرم فعال می باشند. در Channel Type نیز ارتباطات سبز رنگ برقرار می باشند.

****

Test Link

اولین مرحله برای برقراری ارتباط با دستگاه حضور و غیاب یا ژتون، Test Link یا تست ارتباط است با double click روی این گزینه در فرم به این متد از برنامه وارد می شوید:

```
//-----------------------------------//
// tests connection link between pc &//
// selected device.                  //
// fill_link_params first saves link //
// parameters in struct cbf.         //
// see link paremeters in cbf struct.//
```

```
// _xDLL.testLink passes function    //
// params cbf, playBeep(bool) & stats//
// for returning device status.      //
// if playBeep = True then after     //
// receiving cmd, device will play a //
// beep & then sends its status to pc//
// return codes:                     //
// rc = 0 -> device test = ok & its  //
//           status is in stats.     //
// if return code = 0 then status is //
// returned in stats as below:       //
//
// [0..3]   : Serial no.           [unpacked bcd].}
// [4..6]   : Software edition no. [packed bcd].  }
// [7..8]   : System Code#.        [bin].         }
// [9..16]  : ID.                  [unpacked bcd].}
// [17..18] : [NU]                 [bin].         }
// [19..20] : Tail Ofs.            [bin].         }
// [21..21] : Tail Seg.            [bin].         }
// [22..24] : Head.                [bin].         }
// [25..25] : Calendar type.       [bin].         }
//            0 -> Christ,  1 -> Solar.           }
// [26..31] : date+time.[YYMMDDWkHHMM] [bin].     }
// [32..32] : Language.            [binary].      }
//            0:Eng, 1:Farsi, 2:Arabic, 3:Turkish }
// [33..33] : mem percent.         [binary].      }
// [34..36] : Group1.              [unpacked bcd].}
// [37..39] : Group2.              [unpacked bcd].}
// [40..40] : Reader type.         [binary].      }
//            1:Punch, 2:Barcode, 3:Magnet, 4:Contactless}
//            5: mifare.
// [41..41] : Reader status.       [binary].      }
//            0:Active, 1:InActive.               }
// [42..43] : Allowed Cards#.      [binary].      }
// [44..45] : Records#. (msb first)[binary].      }
// [46..46] : record type.         [binary].      }
// [47..47] : card digs#.          [binary].      }
// [48..48] : card saved len.      [binary].      }
// [49..49] : memory status.       [binary].      }
//            0:Good, 96:Bad                      }
// [88]     : mv unit type, 0bh = mv1200, 41h=lite}
// [89..92] : firmware.                           }
//                                   //
// other error codes:                //
// ERR_LINK_TO = 1001 (no response)  //
// ERR_LINK_ERR = 1002 (chksum error)//
// ERR_BAUD_SET = 1003 (baud adjust error)//
// ERR_BAD_MODEM_ID = 1004 (wrong modem id)//
//                                   //
// at the end of function, say_result//
// shows return code.                //
//-----------------------------------//
private void btnTestLink_Click(object sender, EventArgs e)
{
  short rc=0;
  byte[] stats = new byte[200];
  if (!fill_link_params())
```

```
        return;

      show_status("Getting Status", "");
      //88-09-12 if (cbPlayBeep.Checked)
        rc = _xDLL.testLink(cbf, cbPlayBeep.Checked, ref stats);
      say_result(rc);

    }

   private void testDLLFM_Activated(object sender, EventArgs e)
    {
      /*
      tbIoNo.Text      = "1";
      tbBlkLen.Text    = "4000";
      tbMdmID.Text     = "ANI";
      rbPw1410.Checked = true;
      rbCom1.Checked   = true;
      rbRs232.Checked  = true;
      rb115200.Checked = true;
      rbBaudAdjust.Checked = true;
      rbPW1xxx.Checked = true;
      seTimeout.Value = 2000;
      cbClearData.Checked = false;
      cbPlayBeep.Checked = true;

       */
    }
//-----------------------------------//
```

cbf یک structure (متغیر ورودی) است که با توجه به انتخاب تنظیمات کاربر روی فرم اصلی، این structure پر می شود. در این تابع، rc یک متغیر local است که نتیجه متد فوق (انجام شدن یا نشدن تست ارتباط) در آن ریخته می شود. درصورتیکه این مقدار صفر برگرداند یعنی تست ارتباط با موفقیت انجام شده در غیر اینصورت خیر.

تمام اطلاعات دریافت شده از دستگاه (نوع دستگاه، ویرایش دستگاه، تاریخ و زمان دستگاه، سریال دستگاه،...) به متغیر stats ریخته می شود.

****

Set DateTime

گزینه بعدی Set DateTime یا ارسال زمان است. با انتخاب این مورد به متد زیر وارد می شوید:

```
//-----------------------------------------------//
    private void btnSendTime_Click(object sender, EventArgs e)
    {
      short rc = 0;
      if (!fill_link_params())
        return;
```

```
      show_status("Setting Date & Time", "");
//      fill_link_params();
      rc = _xDLL.sendTime(cbf);
      say_result(rc);
    }
```

cbf یک structure است که با توجه به انتخاب تنظیمات کاربر روی فرم اصلی، این structure پر می شود. در این تابع، rc یک متغیر local است که نتیجه متد فوق (ارسال یا عدم ارسال زمان) در آن ریخته می شود.

برای ارسال زمان از زمان pc استفاده می شود بنابراین باید از صحت زمان pc اطمینان حاصل شود.

****

Send Alarms

در گزینه بعد Send Alarms یا ارسال زمانهای آژیر است.

```
//-------------------------------------------------//
    private void btnSendTime_Click(object sender, EventArgs e)
    {
      short rc = 0;
      if (!fill_link_params())
        return;
      show_status("Setting Date & Time", "");
//      fill_link_params();
      rc = _xDLL.sendTime(cbf);
      say_result(rc);
    }

    void put_ALARM(ref NetTypes.ALARM_RECORD_TYPE alarm, byte dOWk, ushort
start, ushort len)
    {
      alarm.dOWk = dOWk;
      alarm.start = start;
      alarm.len = len;
    }

    private void btnSendAlarms_Click(object sender, EventArgs e)
    {
      short rc = 0;
      NetTypes.ALARM_RECORD_TYPE[] alarms = new
NetTypes.ALARM_RECORD_TYPE[7];
//10 alarms for each day. first day is sat.

      if (!fill_link_params())
        return;

      put_ALARM(ref alarms[0], 1, 1 * 60, 11);  //sun
      put_ALARM(ref alarms[1], 2, 2 * 60, 12);  //mon
      put_ALARM(ref alarms[2], 3, 3 * 60, 13);  //tue
      put_ALARM(ref alarms[3], 4, 4 * 60, 14);  //wed
```

```
        put_ALARM(ref alarms[4], 5, 5 * 60, 15);  //thu
        put_ALARM(ref alarms[5], 6, 6 * 60, 16);  //fri
        put_ALARM(ref alarms[6], 7, 7 * 60+54, 17);  //sat

        //put_ALARM(ref alarms[0], 0, 0, 0);    //clear all alarms.

        show_status("Sending Alarms", "");
////        fill_link_params();

        rc = _xDLL.sendAlarms(cbf, alarms);
        say_result(rc);

    }
    //-----------------------------------//
```

در پارامترهای فوق، alarms[0] یک structure است که به معنای اولین خانه از آرایه alarms می باشد ، 1 به معنای مشخص شدن روز هفته، 60*1 به معنای زمان فعال شدن آژیر (ساعت 1 که برای تبدیل به دقیقه در 60 ضرب می شود و درصورت نیاز با مقدار دقیقه جمع می شود)، و متغیر آخر ۱۱ مدت زمان کشیدن آژیر است.

****

Key Cards

گزینه بعد برای Key Cards یا ارسال کارتهای خاص استفاده می شود. در این قسمت وضعیت کلیدهای دستگاه ازلحاظ فعال یا غیر فعال بودن، با رمز یا بدون رمز بودن، تعریف کارت خاص برای کلیدها، با رمز یا بدون رمز بودن و با اثرانگشت یا بدون اثرانگشت بودن کارت استفاده می شود.

```
//-----------------------------------//
    private void btnKeyCards_Click(object sender, EventArgs e)
    {
      short rc = 0;
      NetTypes.KEY_CARD_TYPE[] kCards = new NetTypes.KEY_CARD_TYPE[21];


      if (!fill_link_params())
        return;

      show_status("Setting Key Cards", "");
//      fill_link_params();

      //0,1,2,3,4,5,6,7,8,9,F1,F2,F3,F4,Pgm,up,down,esc,cr

      fill_key_card(ref kCards[0],  0, 0x07); //0
      fill_key_card(ref kCards[1],  0, 0x01); //1
      fill_key_card(ref kCards[2], 0, 0x02); //2
      fill_key_card(ref kCards[3], 0, 0x04); //3
      fill_key_card(ref kCards[4], 0, 0x07); //4
      fill_key_card(ref kCards[5], 0, 0x07); //5
      fill_key_card(ref kCards[6], 0, 0x07); //6
```

```
        fill_key_card(ref kCards[7], 0, 0x07); //7
        fill_key_card(ref kCards[8], 0, 0x07); //8
        fill_key_card(ref kCards[9], 0, 0x07); //9
        fill_key_card(ref kCards[10], 0, 0x0e); //F1
        fill_key_card(ref kCards[11], 0, 0x07); //F2
        fill_key_card(ref kCards[12], 0, 0x07); //F3
        fill_key_card(ref kCards[13], 0, 0x07); //F4
        fill_key_card(ref kCards[14], 0, 0x07); //Pgm
        fill_key_card(ref kCards[15], 0, 0x07); //Up
        fill_key_card(ref kCards[16], 0, 0x07); //Down
        fill_key_card(ref kCards[17], 0, 0x07); //Esc
        fill_key_card(ref kCards[18], 0, 0x07); //Cr
        fill_key_card(ref kCards[19], 0, 0x07); //JT
        fill_key_card(ref kCards[20], 0, 0x07); //JT

        rc = _xDLL.send14xxKeyCards(cbf, kCards);

        say_result(rc);
    }
    //-----------------------------------//
```

در این متد kCards[0] یک structure است که به معنای اولین خانه از آرایه kCards می باشد، 0 در اینجا شماره کارت خاص است که برای این کلید تعریف می شود (شماره کارت عیناً وارد می شود). 0x07 برای تعیین وضعیت کلید و کارت خاص استفاده می شود و از الگوی زیر پیروی می کند:

bit0:act , bit1:cardWPass , bit2:keyWPass , bit3=1->with Finger.

به عنوان مثال در اینجا:

0x07 →0000 0111

bit0:1 , bit1:1 , bit2:1 , bit3=0

kCards[19] و kCards[20] مربوط به دستگاه ژتون هستند که فعلاً کاربرد ندارند.

****

DayLight

قسمت بعدگزینه DayLight یا ارسال زمانهای تغییر ساعت است.

```
//-----------------------------------//
    private void btnDayLight_Click(object sender, EventArgs e)
    {
```

```
      short rc = 0;
      byte advMM,advDD,decMM,decDD;
      ushort minutes;

      if (!fill_link_params())
        return;

      show_status("Setting Daylight saving", "");
//      fill_link_params();
      cbf.dateType = NetConsts.DATE_CHRIST;  // DATE_SOLAR;      //0=christ,
1=solar.

      advMM = 1;                          //adv in 13xx/01/01
      advDD = 1;                          //
      decMM = 6;                          //dec in 13xx/06/31
      decDD = 31;                         //
      minutes = 60;                       //one hour.

      rc = _xDLL.sendDayLight(cbf, advMM, advDD, decMM, decDD, minutes);
      say_result(rc);
    }
```

با کلیک راست برروی DATE_SOLAR و انتخاب go to definition ، می توان نوع تاریخ را شمسی یا میلادی انتخاب کرد.

برای انتخاب تاریخ میلادی بجای DATE_SOLAR، DATE_CHRIST گذاشته می شود.

```
public const byte DATE_CHRIST = 0;
public const byte DATE_SOLAR = 1;
```

همچنین در ادامه تابع می توان تاریخ جلو و عقب بردن ساعت و میزان تغییر زمان را با توجه به توضیحات مشخص کرد.

****

Get IOs

گزینه Get IOs یا دریافت اطلاعات، قسمت بعدی برنامه است. با انتخاب این گزینه متد زیر مشاهده می شود:

```
//-----------------------------------//
    private void btnGetIOs_Click(object sender, EventArgs e)
    {
      ushort records=0;
      string bonesFN = "";
      short rc = 0;
      NetTypes.IN_OUT_RECORD_TYPE[] recs = new
NetTypes.IN_OUT_RECORD_TYPE[10000]; //for 10000 records.
      //each record of recs:
      //public UInt32 cardNo;              //
      //public DATE_TYPE xDate;            //year,month,day.
      //public byte hh;                    //hour
      //public byte mn;                    //minutes
```

```
      //public byte ss;                    //seconds
      //public byte ioType;                //in/out type.
      //public byte almosana;              //number of card's "almosana".
      //public byte flags;                 //bit0=1(with finger), bit1=1(with
reader2), bit2=1(without card), bit3=1(with Identify).

      if (!fill_link_params())
        return;

      show_status("Getting Information", "");
      string fPath = @"C:\xLink\Bones\";

//      fill_link_params();
      cbf.dateType = NetConsts.DATE_SOLAR;    //print dates in solar.(text
file)
      //////cbf.textFormat = NetConsts.TEXT_NEW_FORMAT;  //new ios.txt
format.
      cbf.textFormat = NetConsts.TEXT_NONE;  //no text file.

//89-03-08      rc = _xDLL.getIOs(cbf, fPath, ref records, ref bonesFN,
tbResult);
      rc = _xDLL.getIOs(cbf, fPath, ref records, ref bonesFN, tbResult, ref
recs);

      say_result(rc);

    }
    //-----------------------------------//
```

در متغیر fPath مسیر ذخیره اطلاعات دریافت شده مشخص می گردد (C:\xLink\Bones\). همچنین امکان انتخاب ساخت فایل IOs.txt با فرمت شماره کارت ۸ رقمی (TEXT_OLD_FORMAT) و شماره کارت ۱۲ رقمی (TEXT_NEW_FORMAT) وجود دارد. با انتخاب TEXT_NONE، فایل text ساخته نمی شود.

پارامترهای ورودی _xDLL.getIOs عبارتند از:

records : تعداد رکوردهای گرفته شده در این متغیر قرار می گیرد.

bonesFN : نام فایل bones ساخته شده در این متغیر قرار می گیرد.

tbResult : نمایش کانتر در هنگام دریافت رکوردها می باشد.

recs : درصورتیکه گزینه فعال شود، رکوردهای دریافت شده در این متغیر ذخیره می شود.

****

Recover IOs

در قسمت بعد Recover IOs یا بازیافت اطلاعات را داریم. در این متد هم مانند دریافت اطلاعات، مسیر ذخیره اطلاعات و همچنین بازه تاریخی بازیابی اطلاعات مشخص می شود.

```
//-----------------------------------//
    private void btnRecover_Click(object sender, EventArgs e)
    {
      NetTypes.DATE_TYPE fromDate, toDate;
      ushort records = 0;
      string bonesFN = "";
      short rc = 0;
      NetTypes.IN_OUT_RECORD_TYPE[] recs = new
NetTypes.IN_OUT_RECORD_TYPE[10000]; //for 10000 records.

      if (!fill_link_params())
        return;

      show_status("Recovering Information", "");
      string fPath = @"C:\xLink\Bones\";

//      fill_link_params();
      cbf.dateType = NetConsts.DATE_SOLAR;      //0=christ, 1=solar.
      cbf.textFormat = NetConsts.TEXT_OLD_FORMAT;  //new ios.txt format.

      /*
      cbf.fromDate = new NetTypes.DATE_TYPE();
      cbf.fromDate.y = 1388;
      cbf.fromDate.m = 6;
      cbf.fromDate.d = 1;

      cbf.toDate = new NetTypes.DATE_TYPE();
      cbf.toDate.y = 1388;
      cbf.toDate.m = 6;
      cbf.toDate.d = 5;
      */
      fromDate = new NetTypes.DATE_TYPE();
      fromDate.y = 1388;
      fromDate.m = 9;
      fromDate.d = 1;

      toDate = new NetTypes.DATE_TYPE();
      toDate.y = 1388;
      toDate.m = 9;
      toDate.d = 11;

//89-03-08      rc = _xDLL.recoverIOs(cbf, fPath, fromDate, toDate, ref
records, ref bonesFN, tbResult);
      rc = _xDLL.recoverIOs(cbf, fPath, fromDate, toDate, ref records, ref
bonesFN, tbResult, ref recs);

      say_result(rc);

    }
    //-----------------------------------//
```

پارامترهای ورودی _xDLL.recoverIOs عبارتند از:

records : تعداد رکوردهای گرفته شده در این متغیر قرار می گیرد.

bonesFN : نام فایل bones ساخته شده در این متغیر قرار می گیرد.

tbResult : نمایش کانتر در هنگام دریافت رکوردها می باشد.

recs : رکوردهای دریافت شده در این متغیر ذخیره می شود.

****

Get Ram

مورد بعدی Get Ram یا دریافت تصویر حافظه می باشد.

```
//-----------------------------------//
    private void btnGetRam_Click(object sender, EventArgs e)
    {
      ushort records = 0;
      string bonesFN = "";
      short rc = 0;

      if (!fill_link_params())
        return;

      show_status("Getting Data Image", "");
      string fPath = @"C:\xLink\Dump\";

//      fill_link_params();
      rc = _xDLL.getImage(cbf, fPath, ref records, ref bonesFN, tbResult);

      say_result(rc);

    }
    //-----------------------------------//
```

در اینجا نیز مسیر ذخیره اطلاعات دریافت شده از حافظه مشخص می شود.

پارامترهای ورودی _xDLL.getImage عبارتند از:

records : تعداد رکوردهای گرفته شده در این متغیر قرار می گیرد.

bonesFN : نام فایل bones ساخته شده در این متغیر قرار می گیرد.

tbResult : نمایش کانتر در هنگام دریافت رکوردها می باشد.

****

Send WoFingers Text

گزینه بعد Send WoFingers Text یا ارسال کارتهای بدون اثرانگشت است.

```
//-----------------------------------//
    private void btnSendWoFngTxt_Click(object sender, EventArgs e)
    {
      short rc = 0;

      if (!fill_link_params())
        return;

      show_status("Sending WO Fingers", "");
      string fName = @"C:\xLink\Cards\WO_FNG_O.TXT";
//      fill_link_params();
      rc = _xDLL.sendWithoutFingers(cbf, fName, tbResult);
      say_result(rc);
    }
    //-----------------------------------//
```

شماره کارتهای مورد نظر که قرار است بدون اثرانگشت خوانده شوند، در فایلی با نام WO_FNG_O.TXT در مسیر مشخص شده در متد فوق وارد شده و سپس به دستگاه ارسال می شود.

به عنوان مثال:

```
// Card no[10] + " " + Password[4]
0000000001 0001
0000000002 0002
0000000003 0003
0000000004 0004
0000000005 0005
```

همانطور که مشاهده می شود در اینجا ۱۰ رقم اول شماره کارت و ۴ رقم بعد اسم رمز کارت می باشد.

****

Clear WoFinger

در ادامه مورد Clear WoFinger یا حذف کارتهای بدون اثرانگشت از دستگاه را داریم.

```
//-----------------------------------//
    private void btnClrWoFng_Click(object sender, EventArgs e)
```

```
    {
      short rc = 0;

      if (!fill_link_params())
        return;

      show_status("Clearing WO Fingers", "");
//      fill_link_params();
      rc = _xDLL.clearWithoutFingers(cbf);
      say_result(rc);
    }
    //-----------------------------------//
```

****

Send Allowed Text

مورد بعد Send Allowed Text یا ارسال کارتهای مجاز- غیرمجاز است.

```
//-----------------------------------//
    private void btnSendAllowedTxt_Click(object sender, EventArgs e)
    {
      bool allowedTypeF;
      short rc = 0;

      if (!fill_link_params())
        return;


      show_status("Sending Allowed Cards", "");
      string fName = @"C:\xLink\Cards\FCARDS_O.TXT";
//      fill_link_params();

      //allowedTypeF = false;              //not allowed cards.
      allowedTypeF = true;               //allowed cards.

      rc = _xDLL.sendAllowedCards(cbf, fName, allowedTypeF, tbResult);
      say_result(rc);
    }
    //-----------------------------------//
```

در قسمت allowedTypeF با عبارت true و false نوع کارت ها از لحاظ مجاز یا غیرمجاز بودن مشخص می شود.

شماره کارتهای مورد نظر که قرار است روی دستگاه نامجاز و یا مجاز شوند، در فایلی با نام FCARDS_O.TXT در مسیر مشخص شده در متد فوق وارد شده و سپس به دستگاه ارسال می شوند.

به عنوان مثال:

```
// Card no[10]
0000000001
```

```
0000000002
0000000003
0000000004
0000000005
```

همانطور که مشاهده می شود در اینجا ۱۰ رقم شماره کارت وارد می شود.

****

Clear Allowed Cards

در ادامه مورد Clear Allowed Cards یا حذف کارتهای مجاز- غیرمجاز می باشد.

```
//-----------------------------------//
    private void btnClearAllowed_Click(object sender, EventArgs e)
    {
      short rc = 0;

      if (!fill_link_params())
        return;

      show_status("Clearing Allowed Cards", "");
//      fill_link_params();
      rc = _xDLL.clearAllowedCards(cbf);
      say_result(rc);
    }
    //-----------------------------------//
```

****

در قسمت دستورات مختص PW (only Pw1xxx) در فرم sample ، پنج دستور مشاهده می شود:

Active Times

دستور اول Active Times یا ارسال زمانهای فعال است.

```
//-----------------------------------//
    private void btnActives_Click(object sender, EventArgs e)
    {
      short rc = 0;
      NetTypes.ACTIVE_RECORD_TYPE[] actives = new
NetTypes.ACTIVE_RECORD_TYPE[7];

      if (!fill_link_params())
        return;


      //5 actives for each day. first day is sat.

      /*
```

```
      put_ACTIVE(ref actives[0], 1, 1 * 60, 1 * 60+1);  //sun
      put_ACTIVE(ref actives[1], 2, 2 * 60, 2 * 60+2);  //mon
      put_ACTIVE(ref actives[2], 3, 3 * 60, 3 * 60+3);  //tue
      put_ACTIVE(ref actives[3], 4, 4 * 60, 4 * 60+4);  //wed
      put_ACTIVE(ref actives[4], 5, 5 * 60, 5 * 60+5);  //thu
      put_ACTIVE(ref actives[5], 6, 6 * 60, 6 * 60+6);  //fri
      put_ACTIVE(ref actives[6], 7, 8 * 60+58, 8 * 60+58);  //sat
      */

      put_ACTIVE(ref actives[0], 0, 0, 0);   //clear all active times.

      show_status("Sending Active Times", "");
//      fill_link_params();

      rc = _xDLL.sendActives(cbf, actives);
      say_result(rc);

    }
    //-----------------------------------//
```

در متد فوق می توان از یکشنبه تا شنبه زمان فعال بودن کارتخوان دستگاه را مشخص کرد. در پارامترهای فوق، actives[0] یک structure است که به معنای اولین خانه از آرایه actives می باشد ، 1 به معنای مشخص شدن روز هفته، 1*60 به معنای زمان آغاز غیرفعال شدن کارتخوان (ساعت 1 که برای تبدیل به دقیقه در 60 ضرب می شود و درصورت نیاز با مقدار دقیقه جمع می شود) و 1*60+1 به معنای زمان پایان غیرفعال شدن کارتخوان می باشد.

با 0 گذاشتن مقادیر فوق (put_ACTIVE(ref actives[0], 0, 0, 0);) زمانهای فعال از روی دستگاه حذف می شود.

****

Set IO Mode

دستور بعدی Set IO Mode یا ارسال مد ورود- خروج است.

```
//-----------------------------------//
    private void btnIOMode_Click(object sender, EventArgs e)
    {
      short rc = 0;
      if (!fill_link_params())
        return;
      show_status("Setting In/Out mode", "");
//      fill_link_params();
      rc = _xDLL.sendIOMode(cbf, IO_INOUT);
//      rc = _xDLL.sendIOMode(cbf, IO_LEAVE1);
      say_result(rc);

    }
    //-----------------------------------//
```

در این متد با کلیک راست برروی IO_INOUT و انتخاب go to definition ، می توان مد ورود خروج دستگاه را انتخاب کرد. مدها بصورت زیر تعریف می شوند:

```
public const byte IO_INOUT = 1; //ورود خروج

public const byte IO_START = 11; //شروع کار

public const byte IO_END = 12; //پایان کار

public const byte IO_INTEGRATION = 15; // پیوستگی

public const byte IO_DELAY = 10; //تاخیر سرویس

public const byte IO_LEAVE1 = 2; // مرخصی ۱

public const byte IO_ LEAVE2 = 3; // مرخصی ۲

public const byte IO_ LEAVE3 = 4; // مرخصی ۳

public const byte IO_ LEAVE4 = 5; // مرخصی ٤

public const byte IO_DUTY1 = 6; // ماموریت ۱

public const byte IO_ DUTY2 = 7; // ماموریت ۲

public const byte IO_ DUTY3 = 8; // ماموریت ۳

public const byte IO_ DUTY4 = 9; // ماموریت ٤
```

****

Send Group

دستور سوم Send Group یا ارسال کد گروه است.

```
//-----------------------------------//
    private void btnGroups_Click(object sender, EventArgs e)
    {
      short rc = 0;
      ushort[] groups = new ushort[5];

      if (!fill_link_params())
        return;

      //all pass
      groups[0] = 0;
      groups[1] = 0;
      groups[2] = 0;
      groups[3] = 0;
      groups[4] = 0;

      show_status("Setting Group codes", "");
```

```
//      fill_link_params();
      rc = _xDLL.sendGroupCodes(cbf, groups);
      say_result(rc);

    }
    //-----------------------------------//
```

در این متد کد گروه دستگاه از اولین تا پنجمین کد گروه تعیین و به دستگاه ارسال می شود. با ۰ گذاشتن کد گروه ها، همه کارت ها با هر کد گروه روی دستگاه خوانده می شود.

****

Set Password

دستور بعد Set Password یا ارسال اسم رمز است.

```
//-----------------------------------//
    private void btnPassword_Click(object sender, EventArgs e)
    {
      short rc = 0;
      if (!fill_link_params())
        return;

      show_status("Setting main password", "");
///      fill_link_params();

//      rc = _xDLL.sendPassword(cbf, "12345678");
      rc = _xDLL.sendPassword(cbf, "00000000");
      say_result(rc);

    }
   //-----------------------------------//
```

اسم رمز ۸ رقمی دستگاه در این متد مشخص شده و به دستگاه ارسال می شود.

****

Send Titles

آخرین دستور Send Titles یا ارسال عناوین می باشد.

```
//-----------------------------------//
    private void btnTitles_Click(object sender, EventArgs e)
    {
      //string s;
      //s. .
      //st
      //ASCIIEncoding
```

```
      short rc = 0;
      NetTypes.TITLE_RECORD_TYPE[] titles = new
NetTypes.TITLE_RECORD_TYPE[13];

      if (!fill_link_params())
        return;

      putTitle(ref titles[0],  1,  0x13, "خروج ورود");  //1
      putTitle(ref titles[1],  11, 0x11, "کار شروع");  //2
      putTitle(ref titles[2],  12, 0x10, "کار پایان");  //3
      putTitle(ref titles[3],  15, 0x11, "پیوستگی");  //4
      putTitle(ref titles[4],  10, 0x11, "سرویس تاخیر");  //5
      putTitle(ref titles[5],  2,  0x17, "مرخصی 1");  //6
      putTitle(ref titles[6],  3,  0x17, "مرخصی 2");  //7
      putTitle(ref titles[7],  4,  0x17, "مرخصی 3");  //8
      putTitle(ref titles[8],  5,  0x17, "مرخصی 4");  //9
      putTitle(ref titles[9],  6,  0x1b, "ماموریت 1");  //10
      putTitle(ref titles[10], 7,  0x1b, "ماموریت 2");  //11
      putTitle(ref titles[11], 8,  0x1b, "ماموریت 3");  //12
      putTitle(ref titles[12], 9,  0x1b, "ماموریت 4");  //13

      show_status("Setting I/O Titles", "");
//      fill_link_params();
      rc = _xDLL.sendTitles(cbf, NetConsts.LANG_FARSI, titles);
      say_result(rc);
    }
    //-----------------------------------//
```

در پارامترهای فوق،titles [0] یک structure است که به معنای اولین خانه از آرایه titles می باشد ، عدد 1 کد عنوان ارسالی (ورود خروج)، 0x13 کد ثابت برای این مد (ورود خروج) و عبارت انتهایی "ورود خروج" عنوانی است که با زدن کلید ۱ روی صفحه نمایش دستگاه نشان داده می شود.

همه دستورات یک short هستند. (۱۶ بیت)

Errorها داخل فایلی به نام net consts است. (ctrl+space)

Cbf ، متغیر ورودی، یک structure است که در آن چند فیلد هست.