راهنمای استفاده از

Eos.Clock.dll

برای برقراری ارتباط و استفاده از

دستگاههای

کنترل دسترسی

لازم به ذکر است برنامه مذکور به زبان C# و تحت net framework. نسخه ۲ نوشته و کامپایل شده است و توسط این نسخه از framework و ویرایش های بالاتر قابل استفاده است.

برای برقراری ارتباط با دستگاه کنترل دسترسی ابتدا بایستی به دستگاه متصل شد به این منظور ابتدا یک ساعت را در کد برنامه تعریف می نماییم و به آن ساعت وصل می شویم که طریقه ساخت ساعت و اتصال به آن به شکل زیر است :

ایجاد یک Connection با استفاده از کلاس ConnectionFactory :

در این حالت با توجه به اینکه طریقه ارتباط با ساعت به چه صورت است از یکی از متدهای زیر استفاده می نماییم :

- ارتباط سریال :

```
public Connection GetSerialConnection(int portNumber, int baudRate,
    Parity parity, int readTimeout,int writeTimeout, int waitBeforeread)
{
    Connection connection = ConnectionFactory.CreateSerialConnection(
        portNumber, baudRate, parity, readTimeout, writeTimeout, waitBeforeread);
    return connection;
}
```

- ارتباط شبکه :

```
public Connection GetTcpIpConnection(string  ip, int port, int waitBeforeRead,
    int readTimeout, int writeTimeOut)
{
    Connection connection = ConnectionFactory.CreateTCPIPConnection(ip, port,
        waitBeforeRead, readTimeout, writeTimeOut);
    return connection;
}
```

شرح پارامترهای استفاده شده به شرح زیر است :

portNumber این پارامتر در ارتباط سریال شماره پورت سریال را مشخص میکند و در ارتباط شبکه شماره پورت ارتباطی شبکه را مشخص می کند.

baudRate پهنای باند در ارتباط مودمی و سریال میباشد .

parity اطلاعات پریتی ارتباط که از طریق ساختار System.Io.Ports.Parity قابل تنظیم است.

readTimeout زمان مجاز برای گرفتن جواب زمان خواندن اطلاعات را در ارتباط نشان می دهد .

writeTimeout زمان مجاز برای گرفتن جواب زمان نوشتن اطلاعات را در ارتباط نشان می دهد .

waitBeforeread مدت زمانی که ارتباط , قبل از هر خواندن اطلاعات بایستی صبر کند را نمایش می دهد.

Ip این پارامتر آدرس ساعت را در ارتباط شبکه مشخص می کند.

پس از ایجاد یک شیء connection نوبت به ایجاد ساعت می رسد که روش آن به شرح زیر است :

```
public AccessControlR2 GetAccessDevice(Connection connection, int source, int dest, EncryptionMode encryptionMode)
{
    var device = new AccessControlR2(connection, source, dest, encryptionMode);
    return device;
}
```

شرح پارامترهای استفاده شده به شرح زیر است :

Connection که شیء ارتباط ساخته شده در قسمت بالا می باشد.
source معمولا با عدد ۱ تنظیم میشود.
Dest شماره تنظیم شده بروی دستگاه میباشد که معمولا ۲۵۵ میباشد.
encryptionMode نوع رمزنگاری ارتباط را مشخص میکند.

بعد از این مراحل بایستی به ساعت وصل شویم و مطمئن شویم که اتصال بدرستی برقرار گردیده است :

- با استفاده از دستور Connect :

```
device.Connect();
if (!device.Connected)
{
    MessageBox.Show(@"error in connection");
    return;
}
```

- با استفاده از دستور TestConnection :

```
if (!device.TestConnection())
{
    MessageBox.Show(@"error in connection");
    return;
}
```

در هر دوی این موارد ارتباط با ساعت برقرار میگردد .

لیست عملیات کنترل دسترسی

```
public bool CreateHolidayTable(AccessControlHolidayTable accessControlHolidayTable);
```

ایجاد یک جدول تعطیلات.جدول تعطیلات حاوی یک کد جدول و یک آرایه دوبعدی برای نگهداری شماره ماه و روزهای هر ماه است.لازم به ذکر است که شماره ماهها و روز ها از صفر شروع میشود.

```
public bool CreateTimeZone(AccessControlTimeZone timeZone);
```

ایجاد یک ناحیه زمانی.یک ناحیه زمانی شامل کد، کد جدول تعطیلات و یک آرایه دو بعدی به ازای هر روز قابل تعریف می باشد. روز های قابل تعریف شامل هفت روز هفته به علاوه ۳ روز خاص می باشد. در هر روز میتوان ۴ بازه زمانی تعریف کرد.

```
public bool CreateUser(AccessControlUser user);
```

تعریف کاربر. هر کاربر شامل شماره کارت، کد، رمز، کد ناحیه زمانی و تاریخ شروع و پایان اعتبار کاربر میباشد.

```
public bool DeleteAllEvents();
```

تمامی ورود و خروج ها و وقایع ثبت شده در ساعت، حذف میشود.

```
public bool DeleteAllHolidayTables();
```

همه جداول تعطیلات از ساعت حذف می شود.

```
public bool DeleteAllTimeZones();
```

همه نواحی زمانی تعریف شده در ساعت حذف می شود.

```
public bool DeleteAllUsers();
```

تمامی کاربران تعریف شده در ساعت حذف می شود.

```
public bool DeleteHolidayTable(byte holidayTableIndex);
```

حذف جدول تعطیلات بر مبنای کد جدول.

```
public bool DeleteTimeZone(byte timeZoneIndex);
```

حذف ناحیه زمانی بر مبنای کد ناحیه زمانی.

```
public bool DeleteUserByCode(long code);
```

حذف کاربر بر مبنای شماره کارت.

```
public bool DeleteUserByIndex(short index);
```

حذف کاربر بر مبنای کد کاربر.

```
public List<AccessControl2Record> Dump(int startAddress, int count, out List<byte[]> badRecords);
```

عملیات تخلیه دستگاه را انجام میدهد.لیستی از کل رویدادهای دستگاه برگشت داده میشود همچنین در صورتی که رکورد خراب مشاهده شود آنرا بصورت داده های خام در `badRecords` قرار میدهد.
هر رکورد دستگاه شامل : نوع دستگاه تولید کننده رویداد(کارت، اثر انگشت، پین و دوربین) و همچنین نوع رکورد (دسترسی داده شد، کاربر موجود نیست، تاریخ اعتبار کاربر معتبر نیست، ...)، همچنین تاریخ و ساعت رویداد، شماره تولید شده توسط ماژول(کارت، اثر انگشت و...) می باشد.

```
public override DateTime GetDateTime();
```

تاریخ و ساعت جاری دستگاه را بر می گرداند.

```
public AccessControlHolidayTable GetHolidayTable(byte index);
```

باز گرداندن جدول تعطیلات بر مبنای کد.

```
public List<byte> GetHolidayTablesList();
```

لیستی از کد جداول تعطیلات موجود در دستگاه را بر می گرداند.

```
public bool GetHolidayTablesMemoryStatus(out int holidayTablesMemoryCapacity, out int usedMemory);
```

وضعیت حافظه (کل حافظه و حافظه استفاده شده) جداول تعطیلات دستگاه را بر می گرداند.

```
public override Record GetRecord();
```

رکورد جاری را بر می گرداند.

```
public AccessControlTimeZone GetTimeZone(byte index);
```

ناحیه زمانی را بر مبنای کد بر می گرداند.

```
public List<byte> GetTimeZonesList();
```

لیستی حاوی کد ناحیه های زمانی موجود در حافظه دستگاه را بر می گرداند.

```
public bool GetTimeZonesMemoryStatus(out int timeZonesMemoryCapacity, out int usedMemory);
```

وضعیت حافظه (کل حافظه و حافظه استفاده شده) ناحیه های زمانی دستگاه را بر می گرداند.

```
public AccessControlUser GetUserDataByCode(long code);
```

کاربر را بر مبنای شماره کارت بر می گرداند.

```
public AccessControlUser GetUserDataByIndex(short index);
```

کاربر را بر مبنای کد کاربر بر می گرداند.

```
public List<AccessControlUserIdData> GetUsersList();
```

لیستی حاوی شماره کارت و کد کاربران دستگاه بر می گرداند.

```
public bool GetUsersMemoryStatus(out int usersMemoryCapacity, out int usedMemory);
```

وضعیت حافظه (کل حافظه و حافظه استفاده شده) کاربران دستگاه را بر می گرداند.

```
public override bool IsEmpty();
```

کنترل خالی بودن حافظه رویدادهای دستگاه

```
public override void NextRecord();
```

اشاره گر خواندن رویدادهای دستگاه را یک واحد افزایش میدهد.

```
public override void SetDateTime(DateTime dateTime);
```

تنظیم تاریخ و ساعت دستگاه.

```
public bool GetEventsMemoryStatus(out int eventsMemoryCapacity, out int usedMemory, out int readPointer, out int writePointer);
```

وضعیت حافظه (کل حافظه و حافظه استفاده شده، اشاره گر خواندن و اشاره گر نوشتن) رویدادهای دستگاه را بر می گرداند.