API Functions List

I2CBridge Board Initialization Routines

```
BYTE U2C_GetDeviceCount();
```

The *U2C_GetDeviceCount* function checks how many I2CBridge devices are currently attached.

Parameters:

None.

Return value:

The function returns the number of the I2CBridge devices detected on current computer.

```
U2C_RESULT U2C_GetSerialNum(
      HANDLE hDevice,
      long* pSerialNum
);
```

The *U2C_GetSerialNum* function retrieves the Serial ID number of the current device. This ID is unique for current I2CBridge device and can help to identify it when using a number of devices simultaneously.

Parameters:

hDevice

Handle to the I2CBridge device to retrieve serial number from. The device have to be opened first, using *U2C_OpenDevice* or *U2C_OpenDeviceBySerialNum* functions.

pSerialNum

Pointer to a long integer variable that will be filled with the device serial number.

Return value:

U2C_SUCCESS
Serial number has been successfully obtained.
U2C_HARDWARE_NOT_FOUND
I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_IsHandleValid (
      HANDLE hDevice
);
```

The *U2C_IsHandleValid* function checks whether the device pointed by *hDevice* handle is currently present

Parameters:

hDevice
Handle to the I2CBridge device that will be checked.

Return value:

U2C_SUCCESS
The device pointed by *hDevice* handle is currently present.
U2C_HARDWARE_NOT_FOUND
I2CBridge device pointed by *hDevice* handle is not found.

```
HANDLE U2C_OpenDevice(
      BYTE nDevice
);
```

The *U2C_OpenDevice* function opens the I2CBridge device.

Parameters:

nDevice
The number of the device to open.

Return value:

If the function succeeds, the return value is a valid handle to the specified device. If the function fails, the return value is INVALID_HANDLE_VALUE. This can happen if the specified device is not present.

```
HANDLE U2C_OpenDeviceBySerialNum(
      long nSerialNum
);
```

The *U2C_OpenDeviceBySerialNum* function opens the I2CBridge device with specified Serial ID number.

Parameters:

nSerialNum
The Serial Id number of the device to open.

Return value:

If the function succeeds, the return value is a valid handle to the specified device. If the function fails, the return value is INVALID_HANDLE_VALUE. This can happen if the specified device is not present.

```
U2C_RESULT U2C_CloseDevice(
      HANDLE hDevice
);
```

The *U2C_CloseDevice* function closes the open device handle.

Parameters:

hDevice

Handle to the I2CBridge device to close.

Return value:

U2C_SUCCESS

The device pointed by *hDevice* handle has been successfully closed.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_GetFirmwareVersion(
      HANDLE hDevice,
      PU2C_VESION_INFO pVersion
);
```

The *U2C_GetFirmwareVersion* function retrieves the version of the firmware currently loaded into I2CBridge device pointed by *hDevice* handle.

Parameters:

hDevice

Handle to the I2CBridge device to obtain firmware version from.

pVersion

Pointer to a long integer variable to be filled with the firmware version number.

.

Return value:

U2C_SUCCESS

The firmware version has been successfully retrieved.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_GetDriverVersion(
      HANDLE hDevice,
      PU2C_VERSION_INFO pVersion
);
```

The *U2C_GetDriverVersion* function retrieves the version of the driver used to communicate with I2CBridge device.

Parameters:

hDevice

Handle to the I2CBridge device to obtain the version of the driver used to communicate with it.

pVersion

Pointer to a long integer variable to be filled with the driver version number.

Return value:

U2C_SUCCESS

The driver version has been successfully retrieved.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_VERSION_INFO U2C_GetDllVersion();
```

The *U2C_GetDllVersion* function retrieves the version of the I2CBridge dynamic link library.

Parameters:

None.

Return value:

I2CBridge dynamic link library version.

I2C High Level Routines

```
U2C_RESULT U2C_Read(
      HANDLE hDevice,
      PU2C_TRANSACTION pTransaction
);
```

The *U2C_Read* function will read up to 256 bytes from the i2c slave device.

Parameters:

hDevice

Handle to the I2CBridge device.

pTransaction

Pointer to the U2C_TRANSACTION structure that will be used during the read transaction.

Before calling the function this structure have to be partially filled:

- *nSlaveDeviceAddress* – will contain the slave i2c device address;
- *nMemoryAddressLength* – will contain the internal address length (in bytes from 0 up to 4). If *nMemoryAddressLength* is equal to 0, no address will be sent to device and repeated i2c start condition won't be generated.

- *MemoryAddress* – will contain the internal address offset.
- *nBufferLength* – will contain the number of bytes that will be read from the slave i2c device.

After successful completion of the read operation *Buffer* member of the structure will be filled by valid data.

Return value:

U2C_SUCCESS

The data has been successfully read.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

U2C_SLAVE_OPENNING_FOR_WRITE_FAILED

U2C_SLAVE_OPENNING_FOR_READ_FAILED

U2C_SENDING_MEMORY_ADDRESS_FAILED

```
U2C_RESULT U2C_Write(
      HANDLE hDevice,
      PU2C_TRANSACTION pTransaction
);
```

The *U2C_Write* function will write up to 256 bytes into the i2c slave device.

Parameters:

hDevice

Handle to the I2CBridge device.

pTransaction

Pointer to the U2C_TRANSACTION structure that will be used during the write transaction.

Before calling the function this structure have to be filled:

- *nSlaveDeviceAddress* – will contain the slave i2c device address;
- *nMemoryAddressLength* – will contain the internal address length (in bytes from 0 up to 4). If *nMemoryAddressLength* is equal to 0, no address will be sent to device.
- *MemoryAddress* – will contain the internal address offset.
- *nBufferLength* – will contain the number of bytes that will be written into the slave i2c device.
- *nBuffer* – will contain the data to be sent.

Return value:

U2C_SUCCESS

The data has been successfully written.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

U2C_SLAVE_OPENNING_FOR_WRITE_FAILED

U2C_SENDING_MEMORY_ADDRESS_FAILED

U2C_SENDING_DATA_FAILED

```
U2C_RESULT U2C_ScanDevices(
      HANDLE hDevice,
      PU2C_SLAVE_ADDR_LIST pList
);
```

The *U2C_ScanDevices* function will scan which slave device addresses are currently occupied by I2C slave devices connected to the bus.

Parameters:

hDevice

Handle to the I2CBridge device.

pTransaction

Pointer to the U2C_SLAVE_ADDR_LIST structure that will be filled with slave device addresses. *nDeviceNumber* member will contain the number of the valid addresses in *List* array.

Return value:

U2C_SUCCESS

Operation has been completed successfully and *pList* is filled with valid data.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

I2C Low Level Routines

```
U2C_RESULT U2C_Start(
      HANDLE hDevice
);
```

The *U2C_Start* function will generate the start condition on the I2C interface.

Parameters:

hDevice

Handle to the I2CBridge device.

Return value:

U2C_SUCCESS

Start condition has been generated.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_RepeatedStart(
      HANDLE hDevice
);
```

The *U2C_Start* function will generate the repeated start condition on the I2C interface.

Parameters:

hDevice

Handle to the I2CBridge device.

Return value:

U2C_SUCCESS

Start condition has been generated.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_Stop(
      HANDLE hDevice
);
```

The *U2C_Stop* function will generate the stop condition on the I2C interface. It can be used also for generation the repeated stop condition.

Parameters:

hDevice

Handle to the I2CBridge device.

Return value:

U2C_SUCCESS

Stop condition has been generated.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_PutByte(
      HANDLE hDevice,
      BYTE Data
);
```

The *U2C_PutByte* function transmits a single byte to the I2C bus. It assumes that the bus is available and that the proper Start Condition has previously been generated. Current function doesn't check acknowledge from the I2C slave device, so it's needed to call *U2C_GetAck* to check acknowledge or to use *U2C_PutByteWithAck* instead of *U2C_PutByte* function. This function can be called several times to implement custom I2C protocol. The function does not release the I2C bus after transmission, so at the end of I2C transaction *U2C_Stop* function has to be called.

Parameters:

hDevice

Handle to the I2CBridge device.

pData

The byte to write to the I2C interface.

Return value:

U2C_SUCCESS
Byte has been transmitted to I2C bus.
U2C_HARDWARE_NOT_FOUND
I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_GetByte(
      HANDLE hDevice,
      BYTE* pData
);
```

The *U2C_GetByte* function reads a single byte from the I2C bus. It assumes that the bus is available, that the proper Start Condition has previously been generated and that the slave device has been properly addressed. Current function doesn't generate acknowledge, so it's needed to call the *U2C_PutAck* function or use *U2C_GetByteWithAck* instead of *U2C_GetByte* function. This function can be called several times to implement custom I2C protocol. The function does not release the I2C bus after transmission, so at the end of I2C transaction *U2C_Stop* function has to be called.

Parameters:

hDevice
Handle to the I2CBridge device.
pData
The pointer to the location that receives the byte read from I2C bus.

Return value:

U2C_SUCCESS
Byte has been received from I2C bus.
U2C_HARDWARE_NOT_FOUND
I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_PutByteWithAck(
      HANDLE hDevice,
      BYTE Data
);
```

The *U2C_PutByteWithAck* function transmits a single byte to the I2C bus and checks for acknowledge from I2C slave device. It assumes that the bus is available and that the proper Start Condition has previously been generated. This function can be called several times to implement custom I2C protocol. The function does not release the I2C bus after transmission, so at the end of I2C transaction *U2C_Stop* function has to be called.

Parameters:

hDevice
Handle to the I2CBridge device.
Data
The byte to write to the I2C interface.

Return value:

U2C_SUCCESS

Byte has been transmitted to I2C bus and I2C slave device has been answered with acknowledge.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

U2C_NO_ACK

I2C slave device doesn't acknowledge the byte.

```
U2C_RESULT U2C_GetByteWithAck(
      HANDLE hDevice,
      BYTE* pData,
      BOOL bAck
);
```

The *U2C_GetByteWithAck* function reads a single byte from the I2C bus and then generates acknowledge or not-acknowledge condition according to the value passed in *bAck* parameter. It assumes that the bus is available, that the proper Start Condition has previously been generated and that the slave device has been properly addressed. This function can be called several times to implement custom I2C protocol. The function does not release the I2C bus after transmission, so at the end of I2C transaction *U2C_Stop* function has to be called.

Parameters:

hDevice

Handle to the I2CBridge device.

pData

The pointer to the location that receives the byte read from I2C bus.

bAck

This parameter determines if acknowledge should be generated after the byte is transmitted. If *bAck* is TRUE – acknowledge is generated, if *bAck* is FALSE – non-acknowledge is generated.

Return value:

U2C_SUCCESS

Byte has been received from I2C bus.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_PutAck(
      HANDLE hDevice,
      BOOL bAck
);
```

The *U2C_PuAck* function reads generates acknowledge or not-acknowledge condition according to the value passed in *bAck* parameter. The function does not release the I2C

bus after transmission, so at the end of I2C transaction *U2C_Stop* function has to be called.

Parameters:

hDevice

Handle to the I2CBridge device.

bAck

This parameter determines if acknowledge should be generated. If *bAck* is TRUE – acknowledge is generated, if *bAck* is FALSE – non-acknowledge is generated.

Return value:

U2C_SUCCESS

Acknowledge / non-acknowledge condition has been successfully generated.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_GetAck(
      HANDLE hDevice
);
```

The *U2C_GetAck* function checks for acknowledge from I2C slave device. The function does not release the I2C bus after transmission, so at the end of I2C transaction *U2C_Stop* function has to be called.

Parameters:

hDevice

Handle to the I2CBridge device.

Return value:

U2C_SUCCESS

I2C slave device has been answered with acknowledge.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

U2C_NO_ACK

I2C slave device doesn't acknowledge.

I2C Wire Level Routines

```
U2C_RESULT U2C_ReadScl(
      HANDLE hDevice,
      U2C_LINE_STATE* pState
);
```

The *U2C_ReadScl* function checks the current state of the SCL line of the I2C interface.

Parameters:

hDevice

Handle to the I2CBridge device.

pState

Pointer to the location that will receive the SCL line state:
LS_RELEASED – if line is released (high)
LS_DROPPED_BY_I2C_BRIDGE – if I2CBridge device has pulled down the line.
LS_DROPPED_BY_SLAVE – if I2C slave device has pulled down the line.

Return value:

U2C_SUCCESS

The line state has been successfully obtained.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_ReadSda(
      HANDLE hDevice,
      U2C_LINE_STATE* pState
);
```

The *U2C_ReadSda* function checks the current state of the SDA line of the I2C interface.

Parameters:

hDevice

Handle to the I2CBridge device.

pState

Pointer to the location that will receive the SDA line state:
LS_RELEASED – if line is released (high)
LS_DROPPED_BY_I2C_BRIDGE – if I2CBridge device has pulled down the line.
LS_DROPPED_BY_SLAVE – if I2C slave device has pulled down the line.

Return value:

U2C_SUCCESS

The line state has been successfully obtained.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_ReleaseScl(
      HANDLE hDevice,
);
```

The *U2C_ReleaseScl* function releases the SCL line of the I2C interface. If the line is not pulled down by I2C slave device, it will be raised.

Parameters:

hDevice

Handle to the I2CBridge device.

Return value:

U2C_SUCCESS

The line state has been successfully released.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_ReleaseSda(
      HANDLE hDevice,
);
```

The *U2C_ReleaseSda* function releases the SDA line of the I2C interface. If the line is not pulled down by I2C slave device, it will be raised.

Parameters:

hDevice

Handle to the I2CBridge device.

Return value:

U2C_SUCCESS

The line state has been successfully released.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_DropScl(
      HANDLE hDevice,
);
```

The *U2C_DropScl* function pulls down the SCL line of the I2C interface

Parameters:

hDevice

Handle to the I2CBridge device.

Return value:

U2C_SUCCESS

The line state has been successfully dropped.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.

```
U2C_RESULT U2C_DropSda(
      HANDLE hDevice,
);
```

The *U2C_DropSda* function pulls down the SCL line of the I2C interface

Parameters:

hDevice

Handle to the I2CBridge device.

Return value:

U2C_SUCCESS

The line state has been successfully dropped.

U2C_HARDWARE_NOT_FOUND

I2CBridge device pointed by *hDevice* handle is not found.