clserkyi – serial port API

Data Book

September 2021 - Rev 6.0

www.kayainstruments.com

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

Table of Contents

KAYA INSTRUMENTS
20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

1 Figures and Tables

1.1 List of Tables

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

2 Revision History

Version	Date	Notes
1.0	05/2017	Initial release Serial port API introduced contains the following features: - Connection to remote serial interface - Opening COM port for serial port - Callback of serial port events
2.0	07/2018	Added function clSerialComPortInitEx() Function clSerialComPortInit() is now deprecated
3.0	06/2020	Document rearrangement
4.0	12/2020	Minor corrections
5.0	04/2021	Improved stability of serial port enumeration and communication used with external clserkyi.dll
6.0	09/2021	Added functions: - clGetSupportedBaudRates() - clGetSerialPortIdentifier() - clSetBaudRate() New section added: Important notes and limitations

Table 1 – Revision History

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

3 Introduction

3.1 Safety precautions

Please take the time to read through the precautions listed below to prevent preventable and unnecessary injuries and damage to you, other personnel, or property. Read these safety instructions carefully before your first use of the product, as these precautions contain safety instructions that must be observed. Be sure to follow this manual to prevent misuse of the product.

⚠ Caution! Read Carefully and do not disregard these instructions.
In the event of a failure, disconnect the power supply Disconnect the power supply immediately and contact our sales personnel for repair. Continuing to use the product in this state may result in a fire or electric shock.
If an unpleasant smell or smoking occurs, disconnect the power supply. Disconnect the power supply immediately! Continuing to use the product in this state may result in a fire or electric shock. After verifying that no smoking is observed, contact our sales personnel for repair.
Do not disassemble, repair or modify the product. It may result in a fire or electric shock due to a circuit shortage or heat generation. Contact our sales personnel before inspection, modification, or repair.
Do not place the product on unstable surfaces. Otherwise, it may drop or fall, resulting in injury to persons or the camera.
Do not use the product if dropped or damaged. Otherwise, a fire or electric shock may occur.
Do not touch the product with metallic objects. Otherwise, a fire or electric shock may occur.
Do not place the product in dusty or humid environments, nor where water may splash. Otherwise, a fire or electric shock may occur.
Do not wet the product or touch it with wet hands. Otherwise, the product may fail or cause a fire, smoking, or electric shock.
Do not touch the gold-plated sections of the connectors on the product. Otherwise, the surface of the connector may be contaminated by sweat or skin oil, resulting in contact failure of a connector, malfunction, fire, or electric shock due to static electricity discharge.
Do not use or place the product in the following locations. ▪ Unventilated areas such as closets or bookshelves. ▪ Near oils, smoke, or steam. ▪ Next to heat sources. ▪ A closed (and not running) car where the temperature becomes high. ▪ Static electricity replete locations ▪ Near water or chemicals. Otherwise, a fire, electric shock, accident, or deformation may occur due to a short circuit or heat generation.
Do not place heavy objects on the product. Otherwise, the product may be damaged.
Be sure to discharge static electricity from the body before touching any sensitive electronic components. The electronic circuits in your computer and the circuits on the ***Iron*** camera and the ***Predator II*** board are sensitive to static electricity and surges. Improper handling may seriously damage the circuits. In addition, do not let your clothing come in contact with the circuit boards or components. Otherwise, the product may be damaged.

3.2 Disclaimer

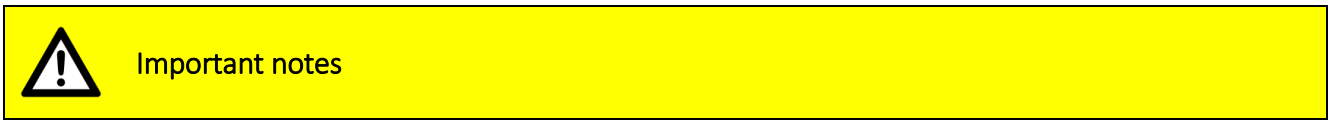
KAYA Instruments assumes no responsibility for any damage that may ensue by using this product for any purpose other than intended, as previously stated. Without detracting from what was previously written, the company takes no responsibility for any damages caused by:

- Earthquake, thunderstrike, natural disasters, a fire caused by use beyond our control, willful and/or accidental misuse and/or use under other abnormal and/or unreasonable conditions.
- Secondary damages caused by the use of this product or its unusable state (business interruption or others).
- Use of this product in any manner that contradicts this manual or malfunctions due to connection to other devices. Damage to this product that is out of our control or failure due to modification
- Accidents and/or third parties that may be involved.

Additionally, KAYA Instruments assumes no responsibility or liability for:

- Erasure or corruption of data caused by the use of this product.
- Any consequences or other abnormalities following the use of this product

3.3 Important API Notes and Limitations



KAYA's API should NOT be used from the DllMain function on Windows OS. There are significant limits on what you can safely do at a DLL entry point. See General Best Practices for specific Windows APIs that are unsafe to call in DllMain. If you need anything but the simplest initialization, then do that in an initialization function for the DLL. You can require applications to call the initialization function after DllMain has run and before they call any other functions in the DLL.

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

4 clserkyi API Functions

The purpose of this document is to list and demonstrate the provided functionality of clserkyi API, a high-level API for serial port communication to remote devices using KAYA's standard API and suitable hardware.

4.1 clGetNumSerialPortsEx()

Returns the number of serial ports provided by a connected remote device.

```
int32_t clGetNumSerialPortsEx(
        CAMHANDLE camHandle,
        uint32_t* numSerialPorts);
```

Parameter name	Type	Description
camHandle	CAMHANDLE	Handle for connected remote device
numSerialPorts	uint32_t*	The number of serial ports in this machine supported by a selected remote device

Return value:
CL_ERR_NO_ERR
CL_ERR_INVALID_PTR

4.2 clSerialInitEx()

Initializes the device referred to by serialIndex and returns a pointer to an internal serial reference structure.

```
int32_t clSerialInitEx(
        CAMHANDLE camHandle,
        uint32_t serialIndex,
        hSerRef* serialRefPtr);
```

Parameter name	Type	Description
camHandle	CAMHANDLE	Handle for connected remote device
serialIndex	uint32_t*	A zero-based index value. For n serial ports of the connected remote device, serialIndex has a range of 0 to (n - 1)
serialRefPtr	hSerRef*	A successful call points to a value that contains a pointer to the vendor-specific reference to the current session.

Return value:
CL_ERR_NO_ERR
CL_ERR_PORT_IN_USE
CL_ERR_INVALID_INDEX
CL_ERR_INVALID_PTR

4.3 clFlushPort()

Discards any bytes that are available in the input buffer.

```
int32_t clFlushPort(
        hSerRef serialRef);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port

Return value:

CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE

4.4 clGetNumBytesAvail()

Outputs the number of bytes received at the port specified by serialRef but are not yet read out.

```
int32_t clGetNumBytesAvail(
        hSerRef serialRef,
        uint32_t* numBytes);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port
numBytes	uint32_t*	The number of bytes currently available to be read from the port

Return value:

CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE
CL_ERR_INVALID_PTR

Example code:

Example of reading data from the serial port, if data is available.

```
hSerRef serialRef = 0;
int32_t error = 0;
error = clSerialInit(0, &serialRef);
if(CL_ERR_NO_ERR != error)
{
        printf("clSerialInit has failed with error=%d\n", error);
        return;
}

char buff[256];
uint32_t buffSize = sizeof(buff);
uint32_t numBytes = 0;

for(;;)
```

```
{
        error = clGetNumBytesAvail(serialRef, &numBytes);
        if(CL_ERR_NO_ERR == error &&
                numBytes > 0)
        {
                buffSize = min(sizeof(buff), numBytes);
                error = clSerialRead(serialRef, buff, &buffSize, 0);
        }
}
```

4.5 clSerialClose()

Closes the serial device and cleans up the resources associated with serialRef. Upon return, serialRef is no longer usable.

```
void clSerialClose(
        hSerRef serialRef);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port

Return value:
CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE

4.6 clSerialRead()

Reads numBytes from the serial device referred to by serialRef. Holds for when numBytes bytes are available at the serial port or specified timeout period has passed. Upon success, numBytes are copied into a buffer. In the case of any error, including CL_ERR_TIMEOUT, no data is copied into the buffer.

```
int32_t clSerialRead(
        hSerRef serialRef,
        char* buffer,
        uint32_t* numBytes,
        uint32_t serialTimeout);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port
numBytes	uint32_t*	The number of bytes requested by the caller
serialTimeout	uint32_t	Indicates the timeout, in milliseconds, for reading specified buffer size
buffer	char*	Points to a user-allocated buffer. Upon a successful call, the buffer contains the data read from the serial device. Upon failure, this buffer is not affected. The caller should ensure that buffer is at least numBytes in size

Return value:

CL_ERR_NO_ERR
CL_ERR_TIMEOUT
CL_ERR_INVALID_REFERENCE

Remarks:

1. Users may try reading more data than what is available on the serial port. If not enough data has been received until the timeout has expired, then CL_ERR_TIMEOUT error is returned, and no data is copied into the buffer.

Example code:

Example of reading data from the serial port.

```
hSerRef serialRef = 0;
int32_t error = 0;
error = clSerialInit(0, &serialRef);
if(CL_ERR_NO_ERR != error)
{
        printf("clSerialInit has failed with error=%d\n", error);
        return;
}

char buff[256];
uint32_t buffSize = sizeof(buff);
error = clSerialRead(serialRef, buff, &buffSize, 300);
switch(error)
{
case CL_ERR_NO_ERR:
        // success
        break;
case CL_ERR_TIMEOUT:
        printf("clSerialRead timed out, smaller buffer should be read\n");
        break;
default:
        printf("clSerialRead general error=%d\n", error);
        break;
}
```

4.7 clSerialWrite()

Writes the data in the buffer to the serial device referenced by serialRef.

```
int32_t clSerialWrite(
        hSerRef serialRef,
        char* buffer,
        uint32_t* bufferSize,
        uint32_t serialTimeout);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port
buffer	char*	Contains data to write to the serial port

bufferSize	uint32_t*	Contains the buffer size indicating the maximum number of bytes to be written. Upon a successful call, bufferSize contains the number of bytes written to the serial device
serialTimeout	uint32_t	Indicates the timeout, in milliseconds, for sending specified buffer size

Return value:
CL_ERR_NO_ERR
CL_ERR_TIMEOUT
CL_ERR_INVALID_REFERENCE

Example code:
Example of writing data to the serial port.

```
hSerRef serialRef = 0;
int32_t error = 0;
error = clSerialInit(0, &serialRef);
if(CL_ERR_NO_ERR != error)
{
        printf("clSerialInit has failed with error=%d\n", error);
        return;
}

char buff[] = {0x00, 0x01, 0x02, 0x03, 0x04};
uint32_t buffSize = sizeof(buff);
error = clSerialWrite(serialRef, buff, &buffSize, 300);
if(CL_ERR_NO_ERR != error)
{
        printf("clSerialWrite has failed with error=%d\n", error);
}
```

4.8 clCallbackRegister()

Registers a callback function for serial ports events specified in KYSP_EVENT_ID.

```
int32_t clCallbackRegister(
        hSerRef serialRef,
        clSerialPortCallback callbackFunc,
        void* userContext);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port
callbackFunc	**clSerialPortCallback**	Callback function for serial port events
userContext	void*	User specific context that will be returned upon callback execution

Return value:
CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

Example code:

Example of registering a callback for received data. The callback is invoked whenever there is data available in the serial receive buffer.

```
void KYCLSER_CALLCONV serialCallbackFunc(void* userContext, KYSP_EVENT* pEvent)
{
	int32_t error = 0;
	switch(pEvent->eventId)
	{
	case KYSP_EVENT_ID_RX_DATA_AVAIL:
	{
		KYSP_EVENT_RX_DATA_AVAIL* eventData = (KYSP_EVENT_RX_DATA_AVAIL*)pEvent;
		hSerRef serialRef = eventData->serialRef;

		uint32_t numBytes = 0;
		clGetNumBytesAvail(serialRef, &numBytes);
		const uint32_t BUFF_SIZE = 256;
		char buffer[BUFF_SIZE] = {0};
		while(numBytes > 0)
		{
			uint32_t bufferSize = min(numBytes, BUFF_SIZE);
			error = clSerialRead(serialRef, buffer, &bufferSize, 300);
			switch(error)
			{
			case CL_ERR_NO_ERR:
				// success
				break;
			case CL_ERR_TIMEOUT:
				printf("clSerialRead timed out, smaller buffer should be read");
				break;
			default:
				printf("clSerialRead general error=%d. Exiting callback", error);
				return;
				break;
			}
		// print received data
			for(int i = 0; i < bufferSize; i++)
			{
				putchar(buffer[i]);
			}
			numBytes -= bufferSize;
		}

		break;
	}
	default:
		break;
	}
}

int main(int argc, char* argv[])
{
```

```
        hSerRef serialRef = 0;
        int32_t error = 0;
        CAMHANDLE camHandle = 0;
        ... // connect to grabber and camera

        clSerialInitEx(camHandle, 0, &serialRef);
        clCallbackRegister(serialRef, serialCallbackFunc, nullptr);

        ...
        for(;;){}
}
```

4.9 clCallbackUnregister()

Unregisters a callback function.

```
int32_t clCallbackUnregister(
        hSerRef serialRef,
        clSerialPortCallback callbackFunc);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port
callbackFunc	**clSerialPortCallback**	Callback function for serial port events

Return value:

CL_ERR_NO_ERR

CL_ERR_INVALID_REFERENCE

4.10 clSerialComPortInitEx()

Creates a Virtual COM Port with a specified port number and display name.

```
int32_t clSerialComPortInitEx(
        CAMHANDLE camHandle,
        uint32_t serialIndex,
        uint32_t* portNumber,
        const char* displayName,
        hSerRef* serialRefPtr);
```

Parameter name	Type	Description
camHandle	CAMHANDLE	Handle for connected remote device
serialIndex	uint32_t	A zero-based index value. For n serial ports of the connected remote device, serialIndex ranges from 0 to (n-1).
portNumber	uint32_t*	COM port number to open. If portNumber is 0, then the next available port number will be assigned. Returns actual port number assigned to COM port
displayName	const char*	COM port display name. In case of NULL, a generic name would be assigned
serialRefPtr	hSerRef*	A successful call points to a value that contains a pointer to the vendor-specific reference to the current session.

Return value:

CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE
CL_ERR_INVALID_PTR
CL_ERR_INVALID_INDEX

Remarks:

1. If portNumber value is 0, then the next available port would be assigned.
2. If portNumber value is non 0, then the COM port with the specified number would try to be created. It is up to the user to make sure the specified COM port is not occupied.

4.11 clSerialComPortInit() (DEPRECATED)

Creates a Virtual COM Port with a specified port number and display name.
This function is deprecated. New applications should use function clSerialComPortInitEx().

```
int32_t clSerialComPortInit(
        hSerRef serialRef,
        uint32_t* portNumber,
        const char* displayName);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port
portNumber	uint32_t*	COM port number to open. If portNumber is 0 then next available port number would be assigned. Returns actual port number assigned to COM port
displayName	const char*	COM port display name. In case of NULL a generic name would be assigned

Return value:

CL_ERR_NO_ERR
CL_ERR_INVALID_REFERENCE
CL_ERR_INVALID_PTR

Remarks:

1. If portNumber value is 0, then the next available port would be assigned.
2. If portNumber value is non 0, then the COM port with the specified number would try to be created. It is up to the user to make sure the specified COM port is not occupied.

4.12 clSerialComPortClose()

Close created COM port.

```
int32_t clSerialComPortClose(
        hSerRef serialRef);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port

Return value:

CL_ERR_NO_ERR

CL_ERR_INVALID_REFERENCE

4.13 clGetNumSerialPorts()

Returns the number of serial ports in your system.

```
int32_t clGetNumSerialPorts(
        uint32_t* numSerialPorts);
```

Parameter name	Type	Description
numSerialPorts	uint32_t*	The number of serial ports in this machine supported by this DLL

Return value:

CL_ERR_NO_ERR

4.14 clSerialInit()

Initializes the device referred to by serialIndex and returns a pointer to an internal serial reference structure.

```
int32_t clSerialInit(
        uint32_t serialIndex,
        hSerRef* serialRefPtr);
```

Parameter name	Type	Description
serialIndex	uint32_t	A zero-based index value. For n serial devices in the system supported by this library, serialIndex has a range of 0 to (n - 1)
serialRefPtr	hSerRef*	On a successful call, points to a value that contains a pointer to the vendor-specific reference to the current session

Return value:

CL_ERR_NO_ERR

CL_ERR_PORT_IN_USE

CL_ERR_INVALID_INDEX

CL_ERR_INVALID_PTR

4.15 clGetErrorText()

Converts an error code to error text for display in a dialog box or a standard I/O window.

```
int32_t clGetErrorText(
        int32_t errorCode,
        char* errorText,
        uint32_t* errorTextSize);
```

Parameter name	Type	Description
errorCode	int32_t	The error code is used to find the appropriate error text. Every function in this library returns an error code
errorText	char*	User allocated buffer which contains the NULL-terminated error text on function return
errorTextSize	uint32_t*	On success, contains the number of bytes written into the buffer, including the NULL-termination character. This value should be the size in bytes of the error text buffer passed in. On CL_ERR_BUFFER_TOO_SMALL, contains the size of the buffer needed to write the error text

Return value:

CL_ERR_NO_ERR
CL_ERR_BUFFER_TOO_SMALL
CL_ERR_ERROR_NOT_FOUND
CL_ERR_INVALID_PTR

Example code:

Example of printing error text if clSerialInit function has failed.

```
hSerRef serialRef = 0;
int32_t error = 0;
error = clSerialInit(0, &serialRef);
if(CL_ERR_NO_ERR != error)
{
        uint32_t errorTextSize = 0;
        clGetErrorText(error, NULL, &errorTextSize);
        char* errorText = (char*)malloc(errorTextSize);
        clGetErrorText(error, errorText, &errorTextSize);
        printf("clSerialInit error: '%s'", errorText);
        free(errorText);
}
```

4.16 clGetManufacturerInfo()

Returns the name of the frame grabber manufacturer who created the DLL and the version of the Camera Link specifications with which the DLL complies.

```
int32_t clGetManufacturerInfo(
        char* manufacturerName,
        uint32_t* bufferSize,
        uint32_t* version);
```

Parameter name	Type	Description
manufacturerName	char*	A pointer to a user-allocated buffer into which the function copies the manufacturer name. The returned name is NULL-terminated.
bufferSize	uint32_t*	As an input, this value should be the size of the buffer that is passed. On a successful return, this parameter contains the number of bytes written into the buffer, including the NULL termination character. On CL_ERR_BUFFER_TOO_SMALL, this parameter contains the size of the buffer needed to write the data text.
version	uint32_t*	A constant stating the version of the Camera Link specifications with which this DLL complies.

Return value:

CL_ERR_NO_ERR
CL_ERR_FUNCTION_NOT_FOUND
CL_ERR_BUFFER_TOO_SMALL
CL_ERR_INVALID_PTR

4.17 clSerialPortCallback

Callback function prototype for serial events.

```
typedef void(KYCLSER_CALLCONV *clSerialPortCallback)(void* userContext,
                                                     KYSP_EVENT* pEvent);
```

4.18 clGetSupportedBaudRates

Returns the valid baud rates of the current interface.

```
int32_t clGetSupportedBaudRates(
        hSerRef serialRef,
        uint32_t* baudRates);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port
baudRates	uint32_t*	Bitfield that describes all supported baud rates of the serial port represented by the CL_BAUDRATE constants

Return value:
CL_ERR_NO_ERR
CL_ERR_FUNCTION_NOT_FOUND
CL_ERR_INVALID_REFERENCE

4.19 clGetSerialPortIdentifier

Returns a manufacturer-specific identifier for each serial port in your system.

```
int32_t clGetSerialPortIdentifier(
        uint32_t serialIndex,
        char* portID,
        uint32_t* bufferSize);
```

Parameter name	Type	Description
serialIndex	uint32_t	A zero-based index value. The valid range for serialIndex is 0 to (n–1), where n is the value of numSerialPorts, as returned by clGetNumSerialPorts
portID	char*	Manufacturer-specific identifier for the serial port
bufferSize	uint32_t*	As an input, this value should be the size of the buffer that is passed. On a successful return, this parameter contains the number of bytes written into the buffer, including the NULL termination character. On CL_ERR_BUFFER_TOO_SMALL, this parameter contains the size of the buffer needed to write the data text

Return value:
CL_ERR_NO_ERR
CL_ERR_BUFFER_TOO_SMALL
CL_ERR_INVALID_INDEX
CL_ERR_INVALID_PTR
CL_ERR_FUNCTION_NOT_FOUND

4.20 clSetBaudRate

Sets the baud rate for the serial port of the selected device. Use clGetSupportedBaudRate to determine supported baud rates.

```
int32_t clSetBaudRate(
        hSerRef serialRef,
        uint32_t baudRate);
```

Parameter name	Type	Description
serialRef	hSerRef	Handle to serial port
baudRate	uint32_t	The baud rate you want to use. This parameter expects the values represented by the CL_BAUDRATE constants

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

Return value:

CL_ERR_NO_ERR
CL_ERR_BAUD_RATE_NOT_SUPPORTED
CL_ERR_INVALID_REFERENCE
CL_ERR_FUNCTION_NOT_FOUND

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

5 Enumerations

5.1 KYSP_EVENT_ID

Defines the event type received in callback function.

Parameter name	Type	Description
KYSP_EVENT_ID_RX_DATA_AVAIL	0x1010	Callback event when data is available on serial port

5.2 Error Type Map

Execution of system error and status. Defines the status returned after each function execution. While some error statuses are general, some point to a specific error.

Parameter name	Type	Description
CL_ERR_NO_ERR	0	Function returned successfully
CL_ERR_BUFFER_TOO_SMALL	-10001	User buffer not large enough to hold data
CL_ERR_MANU_DOES_NOT_EXIST	-10002	The requested manufacturer's DLL does not exist on your system
CL_ERR_PORT_IN_USE	-10003	Port is valid but cannot be opened because it is in use
CL_ERR_TIMEOUT	-10004	Operation not completed within the specified timeout period
CL_ERR_INVALID_INDEX	-10005	Not a valid index
CL_ERR_INVALID_REFERENCE	-10006	The serial reference is not valid
CL_ERR_ERROR_NOT_FOUND	-10007	Could not find the error description for this error code
CL_ERR_BAUD_RATE_NOT_SUPPORTED	-10008	Requested baud rate not supported by this interface
CL_ERR_OUT_OF_MEMORY	-10009	The system is out of memory and could not perform the required actions
CL_ERR_REGISTRY_KEY_NOT_FOUND	-10010	DLL location registry key was not found
CL_ERR_INVALID_PTR	-10011	Some parameters' reference is invalid
CL_ERR_UNABLE_TO_LOAD_DLL	-10098	The DLL was unable to load due to a lack of memory or because it does not export all required functions
CL_ERR_FUNCTION_NOT_FOUND	-10099	The function does not exist in the manufacturer's library
CL_ERR_INVALID_OPERATION	-11000	Operation is invalid for specified serial reference

20 HaMesila St., Nesher 3688520, Israel
POB 25004, Haifa 3125001, Israel
Tel:(+972)-72-2723500 Fax:(+972)-72-2723511

6 Structures

6.1 KYSP_EVENT

General serial port event structure.

Structure Field	Type	Description
eventId	KYSP_EVENT_ID	Id of the received event. It defines how to interpret the event structure received in a callback function.

6.2 KYSP_EVENT_RX_DATA_AVAIL

Event structure for specific event id KYSP_EVENT_ID_RX_DATA_AVAIL enumerated by KYSP_EVENT_ID

Structure Field	Type	Description
serialPortEvent	KYSP_EVENT	General event data
serialRef	hSerRef	Handle to serial port