MCSControl

MCS2 Extension

www.smaract.com

SmarAct
PERFECT MOTION

The information given in this document was carefully checked by our team and is constantly updated. Nevertheless, it is not possible to fully exclude the presence of errors. In order to always get the latest information, please contact our technical sales team.

SmarAct GmbH, Schuette-Lanz-Strasse 9, D-26135 Oldenburg
Phone: +49 (0) 441 - 800879-0, Telefax: +49 (0) 441 - 800879-21
Internet: www.smaract.com, E-Mail: info@smaract.com

Document Version: X

TABLE OF CONTENTS

1 *INTRODUCTION*

1.1 Overview

Together with the MCS2 a new library called SmarActCtl was released. The new API is significantly different from the MCSControl, so making existing software which uses the MCSControl compatible with SmarActCtl may be quite labor intensive. To ease the transition, the MCSControl version 3.0.0 includes an extension which allows communicating with MCS2 devices. The extension communicates with the MCS2 by translating all function calls to the new SmarActCtl API.

To use the MCS2 extension the SmarActCtl DLL needeeds to be installed. The SmarActCtl DLL is loaded dynamically when any of the initialization functions are called. Calls to `SA_FindSystems` will then find MCS1 and MCS2 devices and `SA_OpenSystem` will also open both device types. Also, some additional functions were added to support some of the MCS2 functions. See chapter Function Reference for a list of added functions.

NOTICE
When connecting to network MCS2 devices it's recommended to not specify a port in the locator. Since the port is always required for MCS1 devices, a locator without a port indicates a MCS2 device to the library. This allows to skip the device detection routine, which introduces a significant delay when connecting to a device.

Since the MCS2 is quite different from the MCS1, not all MCS1 functions can be replicated exactly. Some known limitations and compatibilty issues are:

- The MCS2 uses a new positioner type, see Positioner Types for more details.
- The MCS2 doesn't differentiate between closed loop speed and referencing speed. Both values are always identical.
- New MCS2 features, like streaming and command groups, cannot be used through MCSControl.
- End effector channels are not supported.
- The Controller Event System is not supported.

1.2 Positioner Types

The MCS2 introduces a new naming scheme called *positioner types*, which replaces the *sensor type* codes. For compatibilty with existing software, all functions from previous library versions still use the old sensor types. If connected to a MCS2 the DLL performs an automatic translation between the type codes. However, not all type codes have a direct counterpart in the other scheme. Therefore, it is also possible to use the native positioner type codes when working with the MCS2. Note that when connected to a MCS1 only sensor types are supported.

To use positioner type codes with a MCS2 the new functions `SA_GetPositionerType_S` and `SA_GetPositionerType_A` functions were added. These functions return the untranslated positioner type reported by the MCS2 device. Since the numeric values of the two schemes do not overlap, setting positioner type codes does not require a new function. All functions which accept a sensor type, e.g., `SA_SetSensorType_S` and `SA_SetSensorType_A`, also support positioner types now.

The MCS2 allows listing of all supported positioner types by the device and reports information about each positioner type. To access this list the functions `SA_GetPositionerTypeList`, `SA_GetPositionerTypeName` and `SA_GetPositionerTypeProperty` were added. For convenience these functions can also be used with the MCS1, but they only access a static sensor list inside the DLL.

For a list and description of all positioner types refer to the MCS2 Positioner Types document.

1.3 Quiet Mode

Another new feature of the MCS2 is the quiet mode. The quiet mode only allows to perform closed-loop movement and reduces the noise that is emitted from the positioners while moving. It is useful in applications where the noise emission is disturbing. The trade-off between the quiet and the normal mode is the higher heat-dissipation of the controller in quiet mode. For this reason the quiet mode is not recommended for continuous operation.

The quiet mode can be enabled using a channel property:

```
SA_STATUS result = SA_SetChannelProperty_S(
    mcsHandle,                                          // system handle
    0,                                                  // channel index
    SA_EPK(SA_GENERAL,SA_QUIET_MODE,SA_OPERATION_MODE),// property key
    SA_ENABLED                                          // property value
);
```

NOTICE

Note that the quiet mode and low vibration mode cannot be used simultanously on the same channel. Enabling one mode, while the other one is already active, will implicitly disable the already active one.

2 *FUNCTION REFERENCE*

2.1 Detailed Function Description

2.1.1 SA_GetPositionerTypeList

Interface:

```
SA_STATUS SA_GetPositionerTypeList(
      SA_INDEX systemIndex,
      unsigned int *typesList,
      unsigned int *typesListSize
);
```

Description:

This function retrieves a list of supported sensor/positioner types. For MCS systems this is a static list of sensor types known by the DLL. For MCS2 devices the positioner type list is read from the connected device and reflects positioner types supported by the firmware. The caller muss pass a pointer to an unsigned int buffer in *typesList* and set *typesListSize* to the size of the buffer. After the call the function has written a list of sensor/positioner types into *typesList* and the number of types into *typesListSize*. If the supplied buffer is too small to contain the list, the buffer will contain no valid content, but *typesListSize* contains the required buffer size.

Parameters:

- *systemIndex* (unsigned int), **input**: Handle to an initialized system.
- *typesList* (unsigned int), **output**: Pointer to a buffer which holds the sensor/positioner type codes after the function has returned.
- *typesListSize* (unsigned int), **input/output**: Specifies the size of *typesList* before the function call. After the function call it holds the number of value written to *typesList*.

Example:

```
unsigned int types[100];
unsigned int typesSize = 100;
SA_STATUS result = SA_GetPositionerTypeList(mcsHandle,types,&typesSize);
```

```
if(result == SA_OK) {
    // types holds sensor/positioner type codes
    // typesSize holds the number of values written to types
}
```

2.1.2 SA_GetPositionerTypeName

Interface:

```
SA_STATUS SA_GetPositionerTypeName(
      SA_INDEX systemIndex,
      unsigned int type,
      char *outBuffer,
      unsigned int *ioBufferSize
);
```

Description:

This function returns a human readable string for the specified sensor/positioner type. The caller must pass a pointer to a char buffer in *outBuffer* and set *ioBufferSize* to the size of the buffer. After the call the function has written the name of the positioner type into *outBuffer* and the number of written characters into *ioBufferSize*. If the supplied buffer is too small to contain the string, the buffer will contain no valid content but *ioBufferSize* contains the required buffer size.

Parameters:

- *systemIndex* (unsigned int), **input**: Handle to an initialized system.
- *type* (unsigned int), **input**: Positioner type code.
- *outBuffer* (char), **output**: Pointer to a buffer which holds the positioner type name after the function has returned.
- *ioBufferSize* (unsigned int), **input/output**: Specifies the size of *outBuffer* before the function call. After the function call it holds the number of characters written to *outBuffer*.

Example:

```
char name[64];
unsigned int size = sizeof(name);
SA_STATUS result = SA_GetPositionerTypeName(mcsHandle, 300, name, &size);
if(result == SA_OK) {
    // name holds sensor/positioner type name
}
```

2.1.3 SA_GetPositionerTypeProperty

Interface:

```
SA_STATUS SA_GetPositionerTypeProperty(
      SA_INDEX systemIndex,
      unsigned int type,
      unsigned int key,
      unsigned int *value
);
```

Description:

This function can be used to retrieve information about sensor/positioner types. The requested positioner type is specified using the *type* parameter. The property is selected by the *key* parameter. Supported properties and their possible values are:

Property Key	Values
SA_MOVEMENT_TYPE_POS_INFO	SA_LINEAR_MOVEMENT_TYPE, SA_ROTATORY_MOVEMENT_TYPE
SA_REFERENCE_TYPE_POS_INFO	SA_NO_REF_TYPE, SA_END_STOP_REF_TYPE, SA_SINGLE_CODED_REF_TYPE, SA_DISTANCE_CODED_REF_TYPE

Parameters:

- *systemIndex* (unsigned int), **input**: Handle to an initialized system.
- *type* (unsigned int), **input**: Positioner type code.
- *key* (unsigned int), **input**: Positioner type info property.
- *value* (unsigned int), **output**: Value of the specified property.

Example:

```
// get movement type for positioner type 300
unsigned int movementType;
SA_STATUS result = SA_GetPositionerTypeProperty(
    mcsHandle,                  // system handle
    300,                        // positioner type
    SA_MOVEMENT_TYPE_POS_INFO,  // property key
    &movementType               // property value
);
```

```
if(result == SA_OK) {
    // movementType holds the requested value
}
```

2.1.4 SA_GetPositionerType_S

Interface:

```
SA_STATUS SA_GetPositionerType_S(
      SA_INDEX systemIndex,
      SA_INDEX channelIndex,
      unsigned int *type
);
```

Description:

This function is similar to the `SA_GetSensorType_S` function, but returns positioner type codes instead of sensor types, when connected to a MCS2 device. The difference between the two type codes is explained in section Positioner Types.

Parameters:

- *systemIndex* (unsigned int), **input**: Handle to an initialized system.
- *channelIndex* (unsigned int), **input**: Selects the channel of the selected system. The index is zero based.
- *type* (unsigned int), **output**: If the call was successful, this parameter holds the type of the sensor.

Example:

```
// get positioner type for second channel
unsigned int posType;
result = SA_GetPositionerType_S(mcsHandle,1,&posType);
```

2.1.5 SA_GetPositionerType_A

Interface:

```
SA_STATUS SA_GetPositionerType_A(
      SA_INDEX systemIndex,
      SA_INDEX channelIndex
);
```

Description:

In contrast to the synchronous version of this function, this function sends a query to a channel to return the positioner type that is currently configured for it (see `SA_SetSensorType_S`). The actual answer must be retrieved via other functions, e.g. `SA_ReceiveNextPacket_A`. The addressed channel will reply with a packet of type `SA_POSITIONER_TYPE_PACKET_TYPE`. The *data1* field will hold the positioner type. When connected to a MCS1 this function will return the sensor type instead.

Parameters:

- *systemIndex* (unsigned int), **input**: Handle to an initialized system.
- *channelIndex* (unsigned int), **input**: Selects the channel of the selected system. The index is zero based.

Example:

```
// request current positioner type
result = SA_GetPositionerType_A(mcsHandle,0);
SA_PACKET packet;
// wait for answer, but not longer than one second
result = SA_ReceiveNextPacket_A(mcsHandle,1000,&packet);
if (result != SA_OK) {
    // handle error
}
else {
    if ((packet.packetType == SA_POSITIONER_TYPE_PACKET_TYPE) &&
        (packet.channelIndex == 0)) {
        // positioner type is in packet.data1
    }
    else {
        // handle packet otherwise
    }
}
```

Sales partner / Contacts

Headquarters

SmarAct GmbH

Schuette-Lanz-Strasse 9
26135 Oldenburg
Germany

T: +49 441 – 800 87 90
Email: info@smaract.com
www.smaract.com

France

SmarAct GmbH

Schuette-Lanz-Strasse 9
26135 Oldenburg
Germany

T: +49 441 – 80 08 79 956
Email: nicoul@smaract.com
www.smaract.com

Israel

Trico Israel Ltd.

P.O.Box 6172
46150 Herzeliya
Israel

T: +972 9 – 950 60 74

www.trico.co.il

Japan

Physix Technology Inc.

Ichikawa-Business-Plaza
4-2-5 Minami-yawata,
Ichikawa-shi
272-0023 Chiba
Japan
T/F: +81 47 – 370 86 00
Email: info@physix-tech.com
www.physix-tech.com

South Korea

SEUM Tronics

Room 502, 534 Seobusaet-gil
Geumcheon-Gu
08505 Seoul
Korea

T: +82 2 868 – 10 02
Email: hslee@seumtronics.com
www.seumtronics.com

USA

SmarAct Inc.

2140 Shattuck Ave., Suite 1103
94704 Berkeley, CA
United States of America

T: +1 415 – 766 90 06
Email: info@smaract.com
www.smaract.com