

---

# **POSITIONING AND CONTOURING CONTROL SYSTEM APCI-8001, APCI-8008 and CPCI-8004**

## **CanBus Interface**

<b>1 Introduction .....</b>	<b>3</b>
<b>2 Functions of the CanBus option .....</b>	<b>3</b>
2.1 Initialisation .....	3
2.2 Functions of the CanBus Interface .....	4
2.3 Using the CAN Interface .....	5
<b>3 Using the ML71 measuring system.....</b>	<b>6</b>
3.1 Version notes .....	6
3.2 Initialising the CAN module for ML71 .....	6
3.3 Resources for ML71 handling.....	7
3.4 The resource "ML71MeasureDataBlock" .....	7
3.4.1 The "index" element of the resource "ML71MeasureDataBlock" .....	8
3.4.2 The "SubIndex" element of the resource "ML71MeasureDataBlock" .....	9
3.5 Note on using the "ML71MeasureDataBlock" .....	9

# 1 Introduction

The APCI-8001 / CPCI-8004 control units can optionally be fitted with a CanBus interface. To use this option, the “U66 (MSM9225B) / U39” module on the CPCI-8004 must be equipped. The CAN bus is connected at connector P2. RWMOS.ELF operating system software which contains the “optionMSM9255” option must also be used. The APCI-8001 / CPCI-8004 can only act as a master and does not have an object directory itself.

## 2 Functions of the CanBus option

### 2.1 Initialisation

Access to the CAN function is via the “universal object interface” of the APCI-8001 / CPCI-8004. The following values for the universal object interface must be applied when using the CanBus interface.

Table 1: Object descriptor elements

Object descriptor element	Value
Handle	Must be initialised with 0 when starting the application or after rebooting the control unit, and is then managed/used by the system. <b>For PCAP programming:</b> After the resource functionality is cleaned, the handles for all elements <b>must</b> be reset to zero.
BusNumber	400
DeviceNumber	8000 0000 hex index/subindex according to Table 2 or DeviceNumber according to Table 3
Index	Parameters of the respective function according to Table 2 or 3.
SubIndex	Parameters of the respective function according to Table 2 or 3. Unless otherwise specified = 0

Further information on the object descriptor elements can be found in the document “Universal Object Interface”.

## 2.2 Functions of the CanBus Interface

Table 2: CanBus interface functions (BusNumber 400)  
for DeviceNumber = 8000 0000hex:

Device Number	Name	Type	Explanation	Index	Subindex
8000 0000	Clean	integer w	All SAP CanBus objects are rejected. The parameter transferred is meaningless.	0	0
8000 0000	StartNode	integer w	The node selected in Value or all nodes (Value = 0) are started (set to Operational Mode).	1	0
8000 0000	StopNode	integer w	The node selected in Value or all nodes (Value = 0) are stopped (set to Pre-Operational Mode).	2	0
8000 0000	EnableSync	integer r/w	Sync is blocked if 0 Sync is enabled if <> 0	10 hex	0
8000 0000	SendSync	integer w	A sync signal is output	11 hex	0
8000 0000	ResetNode	integer w	The node selected in Value or all nodes (Value = 0) is/are reset (set to Initialisation Mode).	81 hex	0
8000 0000	CanBus Enable	integer w	The CanBus function is enabled if a parameter <> 0 or blocked if parameter = 0.	100 hex	0
8000 0000	CanBus Open	integer r/w	The CanBus is opened. This call is required once per system boot or after a CanBus Close.	101 hex	0
8000 0000	CanBus Close	integer w	The CanBus is closed.	102 hex	0
8000 0000	CanBusInit	integer w	The CanBus is initialised. This call is required at least after every CanBusOpen. If the parameter is 500, a baud rate of 500 kbits/s can be set. (Default: 1 Mbit/s)	103 hex	0
8000 0000	CanBusStart	integer w	The CanBus is started. With this command, the CAN bus controller of the APCI-8001 / CPCI-8004 is started.	104 hex	0
8000 0000	CanBusStop	integer w	The CanBus is stopped. With this command, the CAN bus controller of the APCI-8001 / CPCI-8004 is stopped.	105 hex	0

Table 3: CanBus interface functions (BusNumber 400)  
for DeviceNumber not equal to 8000 0000hex:

Device Number	Name	Type	Explanation	Index	Subindex
600h	PP_FC_SDO_RX	integer r/w	Read and write access to the service data objects of CAN subscribers	Can Index	Can Subindex
200h	PP_FC_PDO1_RX	integer w	Describe process for data channel 1	Number of bytes	
300h	PP_FC_PDO2_RX	integer w	Describe process for data channel 2	Number of bytes	
400h	PP_FC_PDO3_RX	integer w	Describe process for data channel 3	Number of bytes	
500h	PP_FC_PDO4_RX	integer w	Describe process for data channel 4	Number of bytes	

The DeviceNumber represents the CAN identifier. The node number (Node-Id) of the device to be addressed must be additionally specified in the least significant 7 bits of the DeviceNumber. If Node-Id is 0, all nodes are addressed.

This list can be extended for the specific user. The driver level remains unaffected by customised extensions. Only the RWMOS.ELF operating system file must be updated.

## 2.3 Using the CAN Interface

- Run CanBus Clean at program start so that any existing access objects are deleted.
- Request CanBusOpen; if not "true", then assign 1 to each of the variables "CanBusOpen", "CanBusInit" and "CanBusStart"
- Enable CanBus function in RWMOS by calling the "CanBusEnable" function with the parameter 1; only then are accesses via the CanBus possible
- Reset one or all of the CAN subscribers with the command "ResetNode". The NodeID is specified in the parameter. A value of 0 means all CAN subscribers are addressed. Each subscriber sends a message: 700hex + NodeID and a data byte with the value 0.
- Start nodes with the command "CanNodeStart".

**Note:** After a ResetNode, a node must first be registered with RWMOS.ELF before it can be started. Where this is not the case, accessing "CanNodeStart" returns the value 1. This may involve a waiting time (possibly of several seconds).

## 3 Using the ML71 measuring system

When using the ML71 measuring system from Hottinger Baldwin Messtechnik GmbH, an implementation exists for recording measuring values via a CAN interface. Measuring values are handled using the resource interface of the APCI-8001 / CPCI-8004.

The measuring values are transferred to the APCI-8001 / CPCI-8004 via the CAN bus. When using CAN-Bus 1 of the ML71, the measuring values can be processed individually or line by line (resource # 10000) or, in conjunction with the scanner functionality, block by block (resource # 10013). When using CAN-Bus 2 of the ML71, only line-by-line processing can be carried out, since the measuring values are not acknowledged via a sync signal here.

### 3.1 Version notes

The RWMOS.ELF operating system software must include the options "optionMSM9255" (CAN support), "optionRESOURCE" and "optionML71" in order to use the function of the ML71 module. This version is only available from V2.5.3.72.

### 3.2 Initialising the CAN module for ML71

The information in Section 2 sometimes applies here. However, the ML71 measuring system does not support the CANopen protocol. Thus, only the following operations are necessary and permitted for the initialisation and commissioning of the CAN bus.

- Run CanBus Clean at program start so that any existing access objects are deleted.
- Request CanBusOpen; if not "true", then assign 1 to each of the variables "CanBusOpen", "CanBusInit" and "CanBusStart"
- Enable the CanBus function in RWMOS by calling the "CanBusEnable" function with the parameter 1; only then are accesses via the CanBus possible
- Read the measuring values using resource # 10000 or using resource numbers 10000 or 10013 in the scanner module (see Table 4).

### 3.3 Resources for ML71 handling

Table 4: List of device numbers for ML71 handling

Dev.#	Name	Type	Explanation	Parameter index [Subindex]
10000	ML71 Measure Value	<i>integer single short int</i>	Measuring value of an ML71 measuring system (data type depending on measuring system settings)	Index of the measuring value channel (0..127)
10001	ML71 Status	<i>integer r</i>	Bit-coded status word according to Table 5	
10013	ML71 Measure Data Block	<i>datablock r</i>	All measuring values accumulated by an ML71 measuring system as a data block for scanning (data type depending on measuring system settings)	Number of measuring values per record (columns) [Max. number of records (lines)]

The resource 10013 can only be used to record measuring values via the scanner interface. However, even with this resource, a read request must first be performed via the resource interface in order to obtain a valid handle. This read request must be successfully called using the “rdOptionInt” function (see “Universal Object Interface” manual, Section 2.1.2.1). The number of measuring value lines already received via CanBus is returned in the “val” parameter. Using this value, it can be seen whether the data transfer via CAN bus is ready.

### 3.4 The resource “ML71MeasureDataBlock”

There are two options for recording the measuring values received via CAN bus using the scanner module. In the first instance, the resources “ML71MeasureValue” can be integrated in the scan record. In this case, the most recently updated measuring value is read by the scanner. This option is available when using CAN-Bus 1 and CAN-Bus 2 of the ML71.

When using CAN-Bus 1 of the ML71, it is also possible to use the resource “ML71MeasureDataBlock” as the scan object. In this case, all the measuring values accumulated since the last scan time are entered in a data block of a fixed size. It is, therefore, possible to record all measuring values without any gaps.

The scanner is programmed in the same way as for scanning a position value, for example.

The special feature of this resource, however, is the layout of the data block recorded, which itself represents a data structure (record). This data structure has the following layout:

Integer number	Integer status			
integer lines	integer columns			
Line 0: Integer or float Measuring value 1	Integer or float Measuring value 2	...		Integer or float Measuring value m (128 max.)
Line 1: Integer or float Measuring value 1	Integer or float Measuring value 2	...		Integer or float Measuring value m (128 max.)
....				
Line n: Integer or float Measuring value 1 (n = maximum 16)	Integer or float Measuring value 2	...		Integer or float Measuring value m (128 max.)

The size of the data block in a scan is always fixed. The number of lines in this data structure (zn) is specified in the object descriptor element "Index" in the less significant 16 bits. The number of columns (an) is specified in the object descriptor element "SubIndex". The content of "Index" and "SubIndex" is described in more detail below.

The number of elements containing valid measuring values may vary from scan element to scan element and is specified in the first element of the data structure ("Number").

The second element in the data block ("Status") is bit-coded and shows, for example, a possible data overrun. The meaning of this register is described in Table 5.

The "Lines" element specifies how many lines have been written on with measuring values in the current block. The "Columns" element specifies how many columns have been written on with measuring values in the current block. The table with the measuring value follows this. A table element always has a word length of 32 bits. The content of the table elements is derived from the ML71 settings and is not specified by RWMOS.ELF. Here, the data formats 32-bit floating point number, 32-bit integer and 16-bit integer are possible.

Table 5: Bit-coding status word in the "ML71MeasureDataBlock"

Bit No.	Name	Function
0		Currently not used
1		Currently not used
2	Data Overrun	A data overrun has taken place; not all data received could be read.
3	Size Change	The number of measuring values in one measuring value line has changed.
4	LineErr	The number of data lines in DataBlock is not sufficient.
5	ColErr	The number of measuring values in one data line in DataBlock is not sufficient (too few columns defined).

### 3.4.1 The "index" element of the resource "ML71MeasureDataBlock"

Information on the size of the measuring value data block is specified in this element. "Index" must contain the number of measuring values to be recorded in one line as a numerical value (128 max.). This is the number of columns in the measuring value table.



### 3.4.2 The “SubIndex” element of the resource “ML71MeasureDataBlock”

The number of lines with measuring value data in the data structure described above is specified in the “Subindex” element (16 max). The number of lines actually written on by RWMOS.ELF is displayed in the 3rd element of the “Lines” data structure.

The size of the scan record is specified by the values entered in “Index” and “SubIndex”. To avoid unnecessary memory requirements in the scanner record and unnecessary data transfer, these values should not be set any higher than necessary.

## 3.5 Note on using the “ML71MeasureDataBlock”

Recording data and transferring the recorded real-time data into the scanner requires computing time in the real-time task of RWMOS.ELF. For this reason, the sampling time should, where possible, not be set to below the default value of 1.28 ms. Further information on this can be found in the Commissioning Manual (CM) under the key word “SampleTime”.

The “ML71MeasureDataBlock” module is used according to the following scheme:

- Initialisation of the CAN module (see Chapter 3.2)
- Initialisation of and read access to the resource “ML71MeasureDataBlock”
- Initialisation of the scanner using the resource “ML71MeasureDataBlock”
- Performance of the scan with the scanner module (as normal).