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

Scanner Interface

1	Introduction	5
2	Using the scanner functions.....	5
2.1	Boards already implemented	5
2.2	Initialising the scanner	5
2.3	Functions of the scanner module	6
2.4	Scan control	7
2.5	Scan trigger output	7
2.6	Definition of the scan records	8
2.6.1	PCAP programming	8
2.6.2	SAP programming.....	8
2.7	Using the scanner functions	9
2.8	PCAP functions for scanner accesses	10
2.8.1	rdScannerBuffer, read scanner buffer	10
2.8.2	rdScannerBufferSize, read scanner buffer size.....	10
2.8.3	rdScannerLsm, read scanner left spool memory.....	10
2.8.4	rdScannerStatus, read scanner status	11

1 Introduction

The scanner functionality of the APCI-8001 / CPCI-8004 can be used to scan and temporarily store process data in real-time. In doing this, the process data is stored cyclically in a scan record. These records can be read out and processed as record arrays. The resource interface is necessary to use this functionality. It is described in the manual „Resource Interface“.

This option can be used only if there is operating system version RWMos.ELF with the options *optionSCANNER* and *optionRESOURCE*.

2 Using the scanner functions

2.1 Boards already implemented

APCI-3120: 16 analog inputs, 8 analog outputs

APCI-3701: 16 inductive transducers

APCI-3003: 4 analog inputs, simultaneous acquisition

APCI-3501: 8 analog outputs

2.2 Initialising the scanner

The following values for the universal object interface must be used when using the scanner module:

Table 1: Object descriptor elements

Object descriptor element	Value
Handle	Must be initialised with 0 when starting the application or after rebooting the control system, and is then managed/used by the system. For PCAP programming: After the scanner functionality is cleaned, the handles for all elements must be reset to zero using the rdwr functionality.
BusNumber	1100
DeviceNumber	0
Index	0, 1, ... Function number for configuring/operating the scanner, according to table 2.
SubIndex	No function

When the DeviceNumber > 0 and the index is 1, the scan objects are declared. The DeviceNumber must be assigned consecutively.

The parameters to be written are returned as second parameters (value) by calling the function wrOptionInt or assigned directly at SAP programming.

For more information on the object descriptor elements, see the manual "Universal Object Interface".

2.3 Functions of the scanner module

Table 2: Functions of the scanner module for device No. 0

No. (index)	Name	Type	Explanation	Return parameter (value)
1	CLEAN	integer w	Reset scanner The value 1 must be returned. For PCAP programming: Immediately after the reset, the handles for all objects used must be reset to zero using the rdwr functionality.	1
2	INIT	integer w	Initialise scanner before start-up. This means, for example, that the SizeOfRecord is calculated and the data buffer is emptied. Caution: By this calling also the variable HW_SCAN_STROBE is reset.	1
3	STARTSTOP	integer r/w	Start or stop scanner, or request status.	1 = Start 0 = Stop
4	STATUS	integer r	Read the status of the scanner The return value of this function is described for the rdScannerStatus function.	
5	SIZEBUFFER	integer r	Request size of the total memory in the scan buffer (in bytes). Default: 100,000 bytes This value can be set using the SZSCANBUFFER environment variable in fwsetup.	
6	TIMEFACTOR	integer r/w	Read/write time factor in scanning time for scanning data Default value: 1	1, 2, ...
7	RECORDSTO SCAN	integer w	Read/write number of records that should be scanned. If 0 is entered here, scanning will be endless. If more records are to be scanned than fit into the data buffer, the scanned data must be read out during the scan process. Default value: 1	0,1, ...
8	RECORDS SCANNED	integer r	Request number of scanned records. By calling the function ScannerInit the value is reset to 0.	
9	SIZEOF RECORD	integer r	Request size of the record to be scanned (in bytes). This value is only available after calling the INIT function.	

No. (index)	Name	Type	Explanation	Return parameter (value)
10	CHECK BUFFER	integer r	Request size of the free memory in the scan buffer (in bytes).	
11	HW_SCAN_ STROBE	integer r/w	By setting of one or several bits in this register, fast hardware inputs can be defined as strobe inputs for the latch process. RWMOS.ELF must have the respecting options that this option can be used. Caution: This variable is reset by a calling of INIT.	bits 0..7
22	FREE BUFFER	integer w	Internal function for the memory administration, it is not considered for the user.	Memory to be released in bytes
64	SYNCPULSE OUT	integer r/w	Only available with special hardware version: This register allows for a scanner-synchronous pulse output to trigger external components (see Chapter 2.5)	bit-coded axis specification for pulse output

2.4 Scan control

By default, the scan is realised in a time-controlled and sampling-synchronous manner. However, it is also possible to realise the scan in an event-controlled way. For this, as the last scan element, the resource WTLSTRB (#101) is defined for a defined axis. In this case, the scan is recorded when a latch-strobe-signal of the respecting axis has been recognised. Also here, the scan is realised sampling-synchronously. The parameter TIMEFACTOR should always be set to 1.

The latch pulses must not be faster than the sampling times. The latch-strobe-signal is always reset during the recording of the scan record.

2.5 Scan trigger output

In a special hardware version and with RWMOS from V2.5.3.78, it is possible to output a hardware trigger signal synchronously to the scan. This signal can be used, for example, to synchronise external components during data acquisition. Depending on the hardware version, this signal may be an RS422 output or a digital 24 V output.

For this, the scanner variable SYNCPULSEOUT (#64) has to be written on by a bit-coded value in which the axes for the pulse output are flagged by set bits.

Example: 3rd axis = pulse output, the value 4 has to be written in the variable SYNCPULSEOUT

Normally, only one output will be prepared for such a purpose. In the corresponding axis, there is no zero-trace signal and no hardware latch available. When using a 24 V digital output, this can be connected in the usual way as well. The level actually output is the result of the disjunction (OR) of the indicated state information.

Through appropriate hardware preparation, it is also possible to have a fast pulse output from the software environment via the resource #64 (see manual "Resource Interface"), e.g. by writing on the resource #64, an RS422 output or, through appropriate hardware preparation, a digital output can be used at once.

Note: The output of the digital outputs is updated only once per sampling interval of the controller (usually 1.28 ms). A fast pulse output allows for a multiple output during the sampling interval.

2.6 Definition of the scan records

2.6.1 PCAP programming

The data to be scanned is defined by function calls with the following data. The elements to be scanned must first be defined via the G3 Resource Interface. The scan record is constructed in the reverse order to that in which the individual elements are defined. All the elements of the G3 resource interface can be scanned.

Table 3: Definition of the scan record

Object descriptor element	Value
BusNumber	1100
AccessType	Input/Output
DateType	(no function)
DeviceNumber	1, 2,
Index	0, 1, ... TIMEFACTOR for this element
SubIndex	Valid handle for an object descriptor from the G3 Resource Interface

Note: The DeviceNumber must not be equal to 0, and must be assigned uniquely. The initialisation of the objects that shall be recorded, is realised with the access method ATAccessInputOutput (= 3).

The contents of the parameter DataType at the object descriptors of the data to be recorded is of no importance. The variable TIMEFACTOR, which is entered into the index, allows scanning the data record only to whole numbered multiples of the recording intervals. With the value 1 the measurement value is recorded in each recording interval. With the value 0, the measurement value is never recorded. As standard, here the value 1 must be entered.

2.6.2 SAP programming

An AT specifier is declared for each piece of data to be scanned.

Example:

```
var ScanListItem_rp:           double AT %MRScannerBus.1.1.0;
var ScanListItem_axst:        integer AT %MDScannerBus.2.1.0;
var ScanListItem_digi:        integer AT %MDScannerBus.3.1.0;
```

The resource to be scanned must then be assigned once to this variable (each time the SAP programme is started) with the help of the ptr operator. This assignment must be specified in the reverse order to that in which the data is stored in the scan record.

Example:

```
ScanListItem_rp      := ptr(G3R_rp_A1_r);      // real position of axis 1
ScanListItem_digi    := ptr(G3R_digi_A1_r);     // digital inputs
ScanListItem_ain_CH0 := ptr(G3R_ain_CH0_r);     // analogue value channel 0
```

Please ensure that the data types used match.

2.7 Using the scanner functions

- Define the required ObjectDescriptor elements
- Define resources to be scanned, execute at least one read operation, so that there is a valid handle.
- Call scanner CLEAN
- Define a list of scan objects
- Program the number of records to be recorded using the variable RECORDSTOSCAN
- Call scanner INIT
- The scan can now be started and stopped again using scanner STARTSTOP
- The scanned data is read out using the PCAP command rdScannerBuffer (see below)
Here, the scanner status can be requested at any time, using rdScannerStatus, for example.

For PCAP programming: If the ResourceClean function is called repeatedly, all handles of the resource object descriptor elements must be reset.

If the ScannerClean function is called repeatedly, all handles of the scanner object descriptor elements must be reset using the rdwr functionality.

2.8 PCAP functions for scanner accesses

2.8.1 rdScannerBuffer, read scanner buffer

DESCRIPTION:	This function copies the current scanner buffer of the APCI-8001 / CPCI-8004 to a storage area of the calling application. The <i>size</i> parameter specifies the number of bytes to be read. The <i>buffer</i> parameter is an indicator of a storage area in the application, which must be at least <i>size</i> bytes.
BORLAND DELPHI:	function rdScannerBuffer (buffer: PChar; size: integer): integer;
C:	int rdScannerBuffer (char *buffer, int size);
VISUAL BASIC:	function rdScannerBuffer (buffer As String, ByVal size As Long)
PARAMETER:	The parameter <i>buffer</i> is a pointer to a buffer of the application. The buffer must be at least <i>size</i> bytes in size. The parameter <i>size</i> specifies the number of bytes to be read.
RETURN VALUE:	Number of bytes that were successfully copied into the <i>buffer</i> storage area. 0 – if there is no data in the scanner -1 – ScannerBuffer is defined to large -2 – System error in the scanner module
NOTE:	The maximum number of bytes that can be read out can be calculated by subtracting the rdScannerBufferSize() and rdScannerLsm() functions described below. Less than the maximum can also be read out. For the subsequent data analysis, it is reasonable to read out always a multiple of the specified record length. Specific <i>rwmos.elf</i> software is required to execute this command. The data structure in which data is written must correspond to the selection of scan objects. The returned data is not aligned with the word limit.

2.8.2 rdScannerBufferSize, read scanner buffer size

DESCRIPTION:	This function returns the current size of the scanner buffer on the APCI-8001 / CPCI-8004. The return value is given in bytes. By default, the buffer size is set to 100,000 bytes. The buffer size can be increased to a maximum of 13 MB, using a flash environment variable.
BORLAND DELPHI:	function rdScannerBufferSize: integer;
C:	int rdScannerBufferSize(void);
VISUAL BASIC:	Function rdScannerBufferSize() As Long
RETURN VALUE:	Buffer size of the scanner buffer, in bytes.
NOTE:	Specific <i>rwmos.elf</i> software is required to execute this command.

2.8.3 rdScannerLsm, read scanner left spool memory

DESCRIPTION:	This function returns the currently free available working memory of the scanner buffer. During entry in the scanner, this value counts down to 0. During the scanner read-out, the value returns to the ScannerBufferSize.
BORLAND DELPHI:	function rdScannerLsm: integer;
C:	int rdScannerLsm(void);
VISUAL BASIC:	Function rdScannerLsm() As Long
RETURN VALUE:	Free available working memory of the scanner buffer, in bytes.
NOTE:	Specific <i>rwmos.elf</i> software is required to execute this command.

2.8.4 rdScannerStatus, read scanner status

DESCRIPTION:	This function returns the current status of the scanner buffer on the APCI-8001 / CPCI-8004.																											
BORLAND DELPHI:	Function rdScannerStatus: integer;																											
C:	int rdScannerStatus(void);																											
VISUAL BASIC:	Function rdScannerStatus() As Long																											
RETURN VALUE:	Bit-coded scanner status. Table: Bit-coded construction of the scanner status word <table><tr><th>Bit No.</th><th>Name</th><th>Function</th></tr><tr><td>0</td><td>empty</td><td>Status flag: scanner is completely empty</td></tr><tr><td>1</td><td>full</td><td>Status flag: scanner is full. The specified number of records has been entered.</td></tr><tr><td>2</td><td>inprocess</td><td>Status flag: the scanner is currently processing data.</td></tr><tr><td>3</td><td>endless</td><td>Status flag: the 'endless' scan operating mode has been selected.</td></tr><tr><td>8</td><td>norecords</td><td>Error flag: no records have been defined. Error during scan list generation.</td></tr><tr><td>9</td><td>overrun</td><td>Error flag: scanner overrun. The scanner buffer is full. For 'endless' scanning, the oldest, last entered, record is rejected. If 'endless' scanning has not been configured, the scan process is stopped and no new records can be recorded anymore</td></tr><tr><td>10</td><td>Config error</td><td>Error flag: An error was detected at configuration. - unknown data type</td></tr><tr><td>11</td><td>Scan Ressource Not Valid</td><td>Error flag: A resource from the scan list is invalid. Possibly, the content of the resource interface was deleted.</td></tr></table>	Bit No.	Name	Function	0	empty	Status flag: scanner is completely empty	1	full	Status flag: scanner is full. The specified number of records has been entered.	2	inprocess	Status flag: the scanner is currently processing data.	3	endless	Status flag: the 'endless' scan operating mode has been selected.	8	norecords	Error flag: no records have been defined. Error during scan list generation.	9	overrun	Error flag: scanner overrun. The scanner buffer is full. For 'endless' scanning, the oldest, last entered, record is rejected. If 'endless' scanning has not been configured, the scan process is stopped and no new records can be recorded anymore	10	Config error	Error flag: An error was detected at configuration. - unknown data type	11	Scan Ressource Not Valid	Error flag: A resource from the scan list is invalid. Possibly, the content of the resource interface was deleted.
Bit No.	Name	Function																										
0	empty	Status flag: scanner is completely empty																										
1	full	Status flag: scanner is full. The specified number of records has been entered.																										
2	inprocess	Status flag: the scanner is currently processing data.																										
3	endless	Status flag: the 'endless' scan operating mode has been selected.																										
8	norecords	Error flag: no records have been defined. Error during scan list generation.																										
9	overrun	Error flag: scanner overrun. The scanner buffer is full. For 'endless' scanning, the oldest, last entered, record is rejected. If 'endless' scanning has not been configured, the scan process is stopped and no new records can be recorded anymore																										
10	Config error	Error flag: An error was detected at configuration. - unknown data type																										
11	Scan Ressource Not Valid	Error flag: A resource from the scan list is invalid. Possibly, the content of the resource interface was deleted.																										
NOTE:	Specific <i>rwmos.elf</i> software is required to execute this command.																											