POSITIONIER- UND BAHNSTEUERUNG APCI-8001 und APCI-8008

CanBus-Interface

Stand: 08.09.2010, CD-ROM V2.53P
Rev. 5/112018

www.addi-data.de

1 Einführung

Die Steuerung APCI-8001 / APCI-8008 kann optional mit einem CanBus-Interface ausgerüstet werden. Der Anschluss des Can-Bus erfolgt am Stecker P2. Des Weiteren muss eine RWMOS.ELF-Betriebssystem-Software verwendet werden, welche die Option „optionMSM9255“ enthält. Die APCI-8001 / APCI-8008 kann nur als Master fungieren und hat selbst kein Objektverzeichnis.

2 Funktionen der Option CanBus

2.1 Initialisierung

Der Zugriff auf die CAN-Funktionalität erfolgt über das „Universelle Object-Interface“ der APCI-8001 / APCI-8008. Folgende Werte für das Universelle Object-Interface sind für die Verwendung des CanBus-Interface zu verwenden:

Tabelle 1: Object-Descriptor-Elemente

Object-Descriptor Element	Wert
Handle	Muss beim Start der Applikation oder nach Reboot der Steuerung mit 0 initialisiert sein und wird dann vom System geführt / verwendet. **Bei PCAP-Programmierung:** Nach einem Clean der Resource-Funktionalität **müssen** die Handles aller Elemente genullt werden.
BusNumber	400
DeviceNumber	8000 0000 hex Index/Subindex laut Tabelle 2 oder DeviceNumber laut Tabelle 3
Index	Parameter zur jeweiligen Funktion laut Tabelle 2 bzw. 3
SubIndex	Parameter zur jeweiligen Funktion laut Tabelle 2 bzw. 3, falls nicht anders angegeben = 0

Weitere Informationen zu den Object-Descriptor-Elementen sind im Dokument „Universelles Objekt-Interface“ zu finden.

2.2 Funktionen des CanBus-Interface

Tabelle 2: Funktionen CanBus-Interface (BusNumber 400) für DeviceNumber = 8000 0000hex:

DeviceNr	Bezeichnung	Typ	Erläuterung	Index	Subindex
8000 0000	Clean	integer w	Alle SAP-CanBus-Objekte werden verworfen. Der übergebene Parameter ist bedeutungslos.	0	0
8000 0000	StartNode	integer w	Der in Value selektierte Node oder alle Nodes (Value = 0) werden gestartet (in den Operational Mode versetzt).	1	0
8000 0000	StopNode	integer w	Der in Value selektierte Node oder alle Nodes (Value = 0) werden gestoppt (in den Pre-Operational Mode versetzt).	2	0
8000 0000	EnableSync	integer r/w	mit 0 wird Sync gesperrt mit <> 0 wird Sync freigegeben	10 hex	0
8000 0000	SendSync	integer w	Ein Sync-Signal wird ausgegeben	11 hex	0
8000 0000	ResetNode	integer w	Der in Value selektierte Node oder alle Nodes (Value = 0) werden zurückgesetzt (in den Initialisation Mode versetzt).	81 hex	0
8000 0000	CanBus Enable	integer w	Die CanBus-Funktionalität wird freigeschaltet mit einem Parameter <> 0 oder gesperrt mit Parameter = 0.	100 hex	0
8000 0000	CanBus Open	integer r/w	Der Can-Bus wird geöffnet. Dieser Aufruf ist einmalig pro Systemboot oder nach einem CanBus Close notwendig.	101 hex	0
8000 0000	CanBus Close	integer w	Der Can-Bus wird geschlossen.	102 hex	0
8000 0000	CanBusInit	integer w	Der Can-Bus wird initialisiert. Dieser Aufruf ist mindestens nach jedem CanBusOpen erforderlich. Mit dem Parameter 500 kann eine Baudrate von 500 kBit/s eingestellt werden. (Default: 1 MBit/s)	103 hex	0
8000 0000	CanBusStart	integer w	Der Can-Bus wird gestartet. Mit diesem Kommando wird der CAN-Bus-Controller der APCI-8001 / APCI-8008 gestartet.	104 hex	0
8000 0000	CanBusStop	integer w	Der Can-Bus wird gestoppt. Mit diesem Kommando wird der CAN-Bus-Controller der APCI-8001 / APCI-8008 gestoppt.	105 hex	0

Tabelle 3: Funktionen CanBus-Interface (Bus-Nummer 400) für DeviceNumber ungleich 8000 0000hex

Device Number	Bezeichnung	Typ	Erläuterung	Index	Subindex
600h	PP_FC_SDO_RX	integer r/w	Lese- oder Schreibzugriff auf die Servicedaten-Objekte von Can-Teilnehmern	Can Index	Can Subindex
200h	PP_FC_PDO1_RX	integer w	Prozess Datenkanal 1 beschreiben	Anzahl Bytes	
300h	PP_FC_PDO2_RX	integer w	Prozess Datenkanal 2 beschreiben	Anzahl Bytes	
400h	PP_FC_PDO3_RX	integer w	Prozess Datenkanal 3 beschreiben	Anzahl Bytes	
500h	PP_FC_PDO4_RX	integer w	Prozess Datenkanal 4 beschreiben	Anzahl Bytes	

Die DeviceNumber repräsentiert den CAN-Identifier. Die Knotennummer (Node-Id) des zu adressierenden Geräts ist jeweils zusätzlich in den niederwertigsten 7 Bit der DeviceNumber anzugeben. Mit der Node-Id 0 werden alle Nodes angesprochen.

Diese Liste kann benutzerspezifisch erweitert werden. Die Treiberebene bleibt bei kundenspezifischen Erweiterungen unberührt; lediglich die Betriebssystem-Datei RWMOS.ELF muss aktualisiert werden.

2.3 Verwendung des CAN-Interface

- CanBus Clean durchführen beim Programmstart, damit evtl. zuvor vorhandene Zugriffsobjekte gelöscht werden
- CanBusOpen abfragen; wenn nicht „true“, dann den Variablen „CanBusOpen“, „CanBusInit“ und „CanBusStart“ jeweils 1 zuweisen
- CanBus-Funktionalität in RWMOS freigeben durch Aufruf der Funktion „CanBusEnable“ mit dem Parameter 1; erst dadurch sind Zugriffe über den CAN-Bus möglich
- Rücksetzen eines oder aller Can-Teilnehmer mit dem Kommando „ResetNode“.
  Im Parameter wird die NodeID angegeben. Mit dem Wert 0 werden alle Can-Teilnehmer angesprochen. Jeder Teilnehmer sendet eine Meldung: 700hex + NodeID und ein Datenbyte mit dem Wert 0.
- Nodes starten mit dem Kommando „CanNodeStart“.

  Achtung: Ein Node muss sich zunächst nach einem ResetNode bei RWMOS.ELF angemeldet haben, bevor er gestartet werden kann. Falls dies nicht der Fall ist, liefert ein Zugriff auf „CanNodeStart“ den Wert 1 zurück. Gegebenenfalls muss hier einige Zeit (evtl. im Sekundenbereich) gewartet werden.

3 Verwendung des Messystems ML71

Für die Verwendung des Messystems ML71 der Firma Hottinger Baldwin Messtechnik GmbH existiert eine Implementierung, um Messwerte per Can-Interface zu erfassen. Das Handling der Messwerte erfolgt mit Hilfe des Ressourcen-Interface der APCI-8001 / APCI-8008.

Die Übertragung der Messwerte zur APCI-8001 / APCI-8008 erfolgt über den CAN-Bus. Bei der Verwendung des CAN-Bus 1 des ML71 können die Messwerte einzeln bzw. zeilenweise (Ressource # 10000) oder in Verbindung mit der Scanner-Funktionalität blockweise (Ressource # 10013) verarbeitet werden.
Bei der Verwendung des CAN-Bus 2 des ML71 ist die Verarbeitung nur zeilenweise möglich, weil hier die Messwerte nicht per Sync-Signal quittiert werden.

3.1 Versionshinweise

Um die Funktionalität des Moduls ML71 verwenden zu können, muss die Betriebssystem-Software RWMOS.ELF die Optionen „optionMSM9255“ (Can-Unterstützung), „optionRESOURCE“ und „optionML71“ enthalten. Diese Version ist erst ab V2.5.3.72 verfügbar.

3.2 Initialisierung des CAN-Moduls für ML71

Hierzu gelten teilweise die Angaben in Kapitel 2. Das Messsystem ML71 unterstützt jedoch nicht das CAN-Open-Protokoll. Somit sind für die Initialisierung und Inbetriebnahme des CAN-Bus nur die nachfolgenden Operationen notwendig und erlaubt.

- CanBus Clean durchführen beim Programmstart, damit evtl. zuvor vorhandene Zugriffsobjekte gelöscht werden
- CanBusOpen abfragen; wenn nicht „true“, dann den Variablen „CanBusOpen“, „CanBusInit“ und „CanBusStart“ jeweils 1 zuweisen
- CanBus-Funktionalität in RWMOS freigeben durch Aufruf der Funktion „CanBusEnable“ mit dem Parameter 1; erst dadurch sind Zugriffe über den CAN-Bus möglich
- Lesen der Messwerte mit Hilfe der Ressource # 10000 bzw. durch Verwendung der Ressourcen-Nummern 10000 oder 10013 im Scanner-Modul (siehe Tabelle 4)

3.3 Ressourcen für das ML71-Handling

Tabelle 4: Liste der Device-Nummern für das ML71-Handling

Dev.#	Bez.	Typ	Erläuterung	Parameter-Index [Subindex]
10000	ML71 Measure Value	***integer single short int***	Messwert eines ML71-Messsystems (Datentyp je nach Einstellungen des Messsystems)	Index des Messwertkanals (0..127)
10001	ML71 Status	***integer r***	bit-codiertes Statuswort laut Tabelle 5	
10013	ML71 Measure Data Block	***datablock r***	Alle aufgelaufenen Messwerte eines ML71-Messsystems als Datenblock für Scan (Datentyp je nach Einstellungen des Messsystems)	Anzahl der Messwerte pro Datensatz (Spalten) [Max. Anzahl der Datensätze (Zeilen)]

Die Ressource 10013 kann nur zur Messwerterfassung per Scanner-Interface verwendet werden. Zuvor muss jedoch auch bei dieser Ressource ein Lesevorgang per Ressourcen-Interface durchgeführt werden, um ein gültiges Handle zu erhalten. Dieser Lesevorgang muss mit der Funktion „rdOptionInt“ erfolgreich aufgerufen werden (siehe Handbuch „Universelles Objekt-Interface“, Kapitel 2.1.2.1). Im Parameter „val“ wird die Anzahl der bereits per Can-Bus empfangenen Messwertzeilen zurückgegeben. Anhand dieses Werts kann erkannt werden, ob sich die Datenübertragung per Can-Bus im betriebsbereiten Zustand befindet.

3.4 Die Ressource „ML71MeasureDataBlock“

Um die per CAN-Bus empfangenen Messwerte mit dem Scanner-Modul aufzuzeichnen, bestehen zwei Möglichkeiten. Zunächst können die Ressourcen „ML71MeasureValue“ in den Scan-Datensatz eingebunden werden. In diesem Fall wird der jeweils zuletzt aktualisierte Messwert vom Scanner gelesen. Diese Möglichkeit besteht bei der Verwendung des CAN-Bus 1 und des CAN-Bus 2 des ML71.
Bei der Verwendung des CAN-Bus 1 des ML71 besteht außerdem die Möglichkeit, als Scan-Objekt die Ressource „ML71MeasureDataBlock“ zu verwenden. In diesem Fall werden alle seit dem letzten Scan-Zeitpunkt aufgelaufenen Messwerte in einem Datenblock mit fester Größe eingetragen. Somit ist es möglich, alle Messwerte lückenlos aufzuzeichnen.
Die Programmierung des Scanners erfolgt analog, wie z.B. beim Scannen eines Positionswerts.
Die Besonderheit bei dieser Ressource ist jedoch der Aufbau des aufgezeichneten Datenblocks, der selbst eine Datenstruktur (Record) darstellt. Der Aufbau dieser Datenstruktur sieht folgendermaßen aus:

Integer Anzahl	Integer Status			
integer Zeilen	integer Spalten			
Zeile 0: Integer oder Float Messwert 1		Integer oder Float Messwert 2	...	Integer oder Float Messwert m (max. 128)
Zeile 1: Integer oder Float Messwert 1		Integer oder Float Messwert 2	...	Integer oder Float Messwert m (max. 128)
....				
Zeile n: Integer oder Float Messwert 1 (n = maximal 16)		Integer oder Float Messwert 2	...	Integer oder Float Messwert m (max. 128)

Die Größe des Datenblocks in einem Scan ist immer fix. Die Anzahl der Zeilen dieser Datenstruktur (zn) wird im Objekt-Descriptor-Element „Index“, in den niederwertigen 16 Bit angegeben. Die Anzahl der Spalten (an) wird im Objekt-Descriptor-Element „SubIndex“ angegeben. Der Inhalt von „Index“ und „SubIndex“ wird nachfolgend näher erläutert.
Die Anzahl der Elemente, welche gültige Messwerte enthalten, kann von Scan-Element zu Scan-Element variieren und ist jeweils im ersten Element der Datenstruktur („Anzahl“) angegeben.
Das zweite Element im Datenblock („Status“) ist bit-codiert und zeigt z.B. einen eventuellen Datenüberlauf an. Die Bedeutung dieses Registers ist in Tabelle 5 beschrieben.
Im Element „Zeilen“ ist angegeben, wie viele Zeilen im aktuellen Block mit Messwerten beschrieben wurden. Im Element „Spalten“ ist angegeben, wie viele Spalten im aktuellen Block mit Messwerten beschrieben wurden. Danach folgt die Tabelle mit den Messwerten. Ein Tabellenelement hat immer eine Wortlänge von 32 Bit. Der Inhalt der Tabellenelemente ergibt sich aus den Einstellungen des ML71 und ist nicht durch RWMOS.ELF vorgegeben. Hier sind die Datenformate 32-Bit-Gleitpunktzahl, 32-Bit-Ganzzahl und 16-Bit-Ganzzahl möglich.

Tabelle 5: Bit-Codierung Status-Wort im „ML71MeasureDataBlock“

Bit-Nr.	Name	Funktion
0		derzeit nicht verwendet
1		derzeit nicht verwendet
2	Data Overrun	Ein Datenüberlauf hat stattgefunden; nicht alle empfangenen Daten konnten ausgelesen werden.
3	Size Change	Die Anzahl der Messwerte in einer Messwertzeile hat sich geändert.
4	LineErr	Die Anzahl der Datenzeilen im DataBlock reicht nicht aus.
5	ColErr	Die Anzahl der Messwerte einer Datenzeile im DataBlock reicht nicht aus (zu wenige Spalten definiert).

3.4.1 Das Element „Index“ der Ressource „ML71MeasureDataBlock“

In diesem Element werden Informationen über die Größe des Messwert-Datenblocks angegeben. „Index“ muss die Anzahl der aufzuzeichnenden Messwerte in einer Zeile als Zahlenwert enthalten (max. 128). Dies ist die Anzahl der Spalten in der Messwert-Tabelle.

3.4.2 Das Element „SubIndex“ der Ressource „ML71MeasureDataBlock“

Im Element „Subindex“ wird die Anzahl der Zeilen mit Messwertdaten in der oben beschriebenen Datenstruktur angegeben (max. 16). Die Anzahl der von RWMOS.ELF tatsächlich beschriebenen Zeilen werden im 3. Element der Datenstruktur „Zeilen“ angezeigt.
Durch die eingetragenen Werte in „Index“ und „SubIndex“ wird die Größe des Scan-Datensatzes spezifiziert. Um einen unnötigen Speicherbedarf im Scanner-Datensatz und unnötigen Datentransfer zu vermeiden, sollten diese Werte nur so groß wie notwendig eingestellt werden.

3.5 Hinweis zur Verwendung des „ML71MeasureDataBlock“

Das Aufzeichnen der Daten und das Übertragen der aufgezeichneten Echtzeitdaten in den Scanner benötigt Rechenzeit in der Echtzeit-Task von RWMOS.ELF. Deshalb sollte die Abtastzeit möglichst nicht unter den Standardwert von 1,28 ms gesetzt werden. Weitere Informationen hierzu finden Sie im Inbetriebnahme-Handbuch IHB unter dem Stichwort „SampleTime“.

Die Verwendung des Moduls „ML71MeasureDataBlock“ erfolgt nach folgendem Schema:

- Initialisierung des CAN-Moduls (siehe Kap. 3.2)
- Initialisierung und Lesezugriff auf die Ressource „ML71MeasureDataBlock“
- Initialisierung des Scanners unter Verwendung der Ressource „ML71MeasureDataBlock“
- Scan mit Scanner-Modul durchführen (wie gewohnt).