Inhaltsverzeichnis

1 Überblick

1.1 Einleitung

Die Bibliothek

1.2 BMC Messsysteme GmbH

BMC Messsysteme GmbH steht für innovative Messtechnik "made in Germany". Vom Sensor bis zur Software bieten wir alle für die Messkette benötigten Komponenten an.

Unsere Hard- und Software ist aufeinander abgestimmt und dadurch besonders

2 Installation

2.1 Installation unter Windows®

Unter Windows® ist die **LIBAD4**

/usr/local/lib/libad.so.4 und **/usr/local/lib/libad.so** an, so dass diese auf **/usr/local/lib/libad.so.4.1.333**

3 Grundlagen

3.1 Einführung

Die von **LIBAD** exportierten Funktionen und die verwendeten Konstanten

4.1.3 ad_discrete_in

Prototype	`int32_t`
	`ad_discrete_in (int32_t adh, int32_t cha,`
	`                int32_t range, uint32_t *data);`

C

```
int32_t adh;
int32_t st;
uint32_t data;

...

  adh = ad_open ("usb-ad");

  st = ad_discrete_in (adh, AD_CHA_TYPE_ANALOG_IN|1,
                       0, &data)

...

  ad_close (adh);
```

Die Funktion **`ad_discrete_in()`** liefert einen Einzelwert des angegebenen Kana_s. Neben der Kana_nummer wird der Funktion noch der Messbereich übergeben998n dem der E8ngangskana_ abgetastet werden soll. Der Messbereich wird für die Eingriwerter( ) den TJ0 d195817 TD-0.0081 Tc012319 Tw[D)-423(i)065(e Funk)-489(ad_discrete_in()

schreg 0x000000 ( )]TJ/TT10 1 Tf reiktionefet

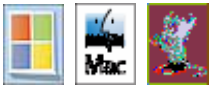
Die Kanalnummer und die Nummer des Messbereichs ist abhängig von der eingesetzten Messhardware und in den entsprechenden Kapiteln dokumentiert (s. "Messsysteme", S. 61).

4.1.4 ad_discrete_in64

Für die Umrechnung eines solchen Werts in einen Spannungswert steht die Funktion **`ad_sample_to_float64()`** zur Verfügung. Die Hilfsfunktion **`ad_analog_in()`** übergibt den Messwert direkt als Spannung.

4.1.8 ad_discrete_outv

4.1.10 ad_sample_to_float64



C

```
int32_t adh;
int32_t st, cha, range;
uint64_t data;

...

  adh = ad_open ("usb-ad");

...

  cha = AD_CHA_TYPE_ANALOG_OUT|1;
  range = 0;

  st = ad_float_to_sample64 (adh, cha, range, 3.2,
                             &data)
  if (st == 0)
    st = ad_discrete_out64 (adh, cha, range, data)

...

  ad_close (adh);
```

4.1.13 ad_analog_in

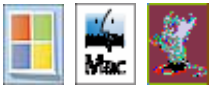
Prototype

```
int32_t
ad_analog_in (int32_t adh, int32_t cha,
              int32_t range, float *volt);
```

Diese Hilfsfunktion ruft **`ad_discrete_in()`** auf und rechnet dann den

Prototype	`int32_t`
	`ad_digital_out (int32_t adh,`
	`int32_t cha, uint32_t data);`

4.1.21 ad_get_version



5 Scanvorgang

5.1 Einführung

Neben der Einzelwertabfrage von Messwerten kann die **LIBAD4** auch einen Scanvorgang starten. Dieser tastet mehrere Eingangskanäle in einem festen Zeitraster ab und liefert die erfassten Messwerte in einem Buffer zurück.

Dabei unterscheidet dif2 re-4.8(i) 34J16.7246 0TD-0.0016[Tc(r)-4.B TwifD4

Mittelwert des Kanals a wird im Verhältnis 1:5 gespeichert (d. h. **`store`** steht auf **`AD_STORE_AVERAGE`** und **`ratio`** auf **`5`**).

Werden alle Kanäle auf **AD_TRG_NEVER** gestellt, dann wird kein Trigger

ü **`samples_per_run`**

Wird von **LIBAD4** zurückgegeben, legt die Anzahl der Messwerte eines Buffers fest, der von **`ad_get_next_run_f()`** zurückgegeben wird.

Nicht verwendete bzw. undokumentierte Elemente der Struktur müssen unbedingt auf `0`

5.3.2 Kanalnummerierung

Die Kanalnummerierung in einem Scan mit CAN Signalen entspricht der Reihen-

C

```
struct ad_scan_cha_desc chav[2];
...
memset (chav, 0, sizeof(chav));

chav[0].cha = AD_CHA_TYPE_ANALOG_IN|1;
chav[0].store = AD_STORE_DISCRETE;
chav[0].ratio = 1;
chav[0].trg_mode = AD_TRG_NONE;

chav[1].cha = AD_CHA_TYPE_ANALOG_IN|3;
chav[1].store = AD_STORE_DISCRETE;
chav[1].ratio = 1;
chav[1].trg_mode = AD_TRG_NONE;
```

Außerdem müssen die globalen Scanparameter in der Struktur **`struct ad_scan_cha_desc`** gesetzt werden. Folgendes Beispiel setzt die Abtastrate auf 1kHz und speichert 500 Messwerte (pro Kanal).

5.5 Kontinuierliche Messung

Neben dem "memory-only"-Scan bietet die **LIBAD4** auch die Möglichkeit eine

C

```
int32_t rc;
struct ad_scan_cha_desc chav[2];
struct ad_scan_desc sd;

...

memset (&chav, 0, sizeof(chav));
memset (&sd, 0, sizeof(sd));

chav[0].cha = AD_CHA_TYPE_ANALOG_IN|1;
chav[0].store = AD_STORE_DISCRETE;
chav[0].ratio = 1;
chav[0].trg_mode = AD_TRG_NONE;

chav[1].cha = AD_CHA_TYPE_A3ratio = 1; chav[0].store = AD_STORE_DISC


...
```

6 MesssCsteme

Ein- bzw. Ausgangskanäle werden in **LIBAD4** durch Kanalnummern spezifiziert. Die Kanalnummer (Integer mit 32Bit) legt neben der eigentlichen Nummer des

6.1.2 Kanalnummern iM3250T

Der erste analoge Eingangskanal eines iM3250T beginnt bei 17. Damit erg2ben sich für die 32 analogen Eingänge folgende Konstanten:

```
#define AI1   (AD_CHA_TYPE_ANALOG_IN|0x0011)
#define AI2   (AD_CHA_TYPE_ANALOG_IN|0x0012)
...
#define AI32  (AD_CHA_TYPE_ANALOG_IN|0x0030)
```

6.1.3 Kanalnummern iM3250

Die Kanalnummern des iM3250 hängen von der Ausbaustufe des Geräts ab. Ist nur eine BPL im Gerät vorhanden, erscheinen die ersten 16 Kanäle von 1 bis 16. Falls beide BPLs eingebaut sind, erscheinen die ersten 16 Kanäle von 17 bis 32. Die zweiten 16 Eingänge sind immer unter den Nummern 33 bis 48 erreichbar.

```
#ifdef BPL1    /* 1 bpl installed /

#define AI1   (AD_CHA_TYPE_ANALOG_IN|0x0001)
#define AI2   (AD_CHA_TYPE_ANALOG_IN|0x0002)
...
#define AI16  (AD_CHA_TYPE_ANALOG_IN|0x0010)

#else          /* 2 bpl's installed /

#define AI1   (AD_CHA_TYPE_ANALOG_IN|0x0011)
#define AI2   (AD_CHA_TYPE_ANALOG_IN|0x0012)
...
#define AI16  (AD_CHA_TYPE_ANALOG_IN|0x0020)

#endif /* BPL1 */

#define AI17  (AD_CHA_TYPE_ANALOG_IN|0x0021)
#define AI18  (AD_CHA_TYPE_ANALOG_IN|0x0022)
...
#define AI32  (AD_CHA_TYPE_ANALOG_IN|0x0030)
```

6.2 PCI-BASE300/1000

Um eine PCI-BASE300/1000 mit der **LIBAD4** zu öffnen, muss an **`ad_open()`** der String "**`pci300`**" übergeben werden. Beim Öffnen des Treibers wird nicht zwischen PCI-BASE300 und PCI-BASE1000 unterschieden.

Mehrere Karten lassen sich durch Angabe der Kartennummer unterscheiden (1. Karte mit "**`pci300:0`**", 2. Karte mit "**`pci300:1`**", usw.).

6.2.1 MAD12/12a/12f/16/16a/16f

Der erste analoge Eingangskanal eines MAD12/12a/12f/16/16a/16f beginnt bei 1. Sobald ein zweites analoges Eingangsmodul auf der PCI-BASE300/1000 gesteckt

Modul	Analog

6.2.3 MCAN

Mess-system	Analog	Kanal-nummer	range (Messber.)	range (Aus-gabebereich)	Digital	Kanal-nummer

```
#define AO1   (AD_CHA_TYPE_ANALOG_OUT|0x0001)
```

6.8 USB-AD / USB-PIO


Um ein USB-AD oder eine USB-PIO mit der **LIBAD4** zu öffnen, muss an **`ad_open()`**

Der erste analoge Eingangskanal eines USB-AD beginnt bei 1. Damit ergeben sich für die 16 Analogeingänge folgende Konstanten:

```
#define AI1   (AD_CHA_TYPE_ANALOG_IN|0x0001)
#define AI2   (AD_CHA_TYPE_ANALOG_IN|0x0002)
...
#define AI16  (AD_CHA_TYPE_ANALOG_IN|0x0010)
```


7 Index

O

P

W

Z