ADwin-32-I/O

32 konfigurierbare digitale Ein-/Ausgänge

Option für ***ADwin***-Meßwerterfassungskarten

Version 1.2

Juli 1998

Inhalt

1 *ADwin-32-I/O*-Option für *ADwin*-Karten

1.1 Funktionsbeschreibung

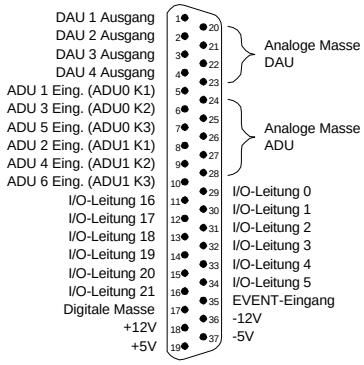
Mit der ***ADwin-32-I/O***-Option werden die fest vorgegebenen digitalen Ein-/Ausgänge der ***ADwin***-Karte ersetzt. Sie erhalten dadurch die Möglichkeit 32 Leitungen in Viererblöcken (sogenannten Nibbles) als digitale Ein- oder Ausgänge zu programmieren.

Die Funktionen und Steckerbelegungen der analogen Ein-/Ausgänge und des Event-Eingangs bleiben unbeeinflußt. Lediglich die Benennung bzw. Zuweisung der Stecker-/Buchsenbelegung für die digitalen Ein- und Ausgänge ändert sich.

1.2 Stecker- und Buchsenbelegungen der ADwin-Karte mit ADwin-32-I/O-Option

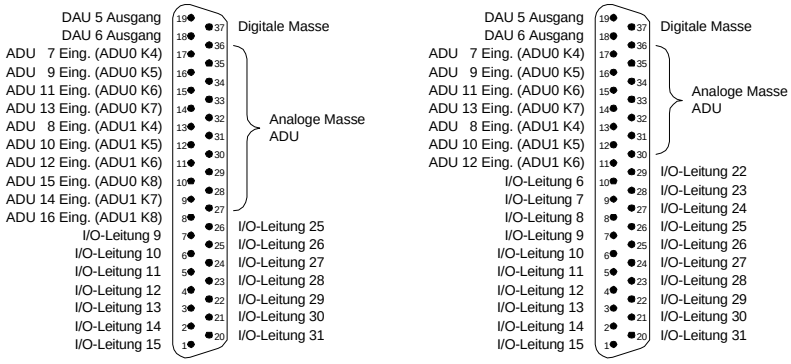
An der Stelle der digitalen Ausgänge befinden sich die I/O-Leitungen 0 bis 15.

An der Stelle der digitalen Eingänge befinden sich die I/O-Leitungen 16 bis 31.



ADwin-32-I/O

Abbildung 1: *ADwin*-Karte mit *ADwin-32-I/O*-Option



ADwin-32-I/O Erw.-Stecker (16 AIN / 26 DIO)

ADwin-32-I/O Erw.-Stecker (13 AIN / 32 DIO)

Abbildung 2: Steckerbelegungen der Erweiterungsstecker

I/O-Leitung:	auf:	Pin-Nr.:
0	***ADwin***-Karte	29
1	***ADwin***-Karte	30
2	***ADwin***-Karte	31
3	***ADwin***-Karte	32
4	***ADwin***-Karte	33
5	***ADwin***-Karte	34
6	Erw.-Stecker	10
7	Erw.-Stecker	9
8	Erw.-Stecker	8
9	Erw.-Stecker	7
10	Erw.-Stecker	6
11	Erw.-Stecker	5
12	Erw.-Stecker	4
13	Erw.-Stecker	3
14	Erw.-Stecker	2
15	Erw.-Stecker	1

I/O-Leitung:	auf:	Pin-Nr.:
16	***ADwin***-Karte	11
17	***ADwin***-Karte	12
18	***ADwin***-Karte	13
19	***ADwin***-Karte	14
20	***ADwin***-Karte	15
21	***ADwin***-Karte	16
22	Erw.-Stecker	29
23	Erw.-Stecker	28
24	Erw.-Stecker	27
25	Erw.-Stecker	26
26	Erw.-Stecker	25
27	Erw.-Stecker	24
28	Erw.-Stecker	23
29	Erw.-Stecker	22
30	Erw.-Stecker	21
31	Erw.-Stecker	20

Tabelle 1: Zuordnung der I/O-Leitungen und der Pin-Nummern

2 Konfiguration der I/O-Leitungen

Nach dem Einschalten sind zunächst alle I/O-Leitungen als Eingänge konfiguriert.

Mit Hilfe des Konfigurationsregisters an Adresse **0E0h** können die I/O-Leitungen in Vierer-blöcken (Nibbles) als Eingänge oder als Ausgänge konfiguriert werden.

Jedes Bit repräsentiert ein Nibble. Somit sind für die 32 zu programmierenden I/O-Leitungen nur die niederwertigsten acht Bit (Bit 0...7) relevant (alle anderen Bits werden ignoriert).

Das niederwertigste Bit (Bit 0) konfiguriert dabei das niederwertigste Nibble (I/O-Leitungen 0...3), das nächsthöhere Bit (Bit 1) konfiguriert das nächsthöhere Nibble (I/O-Leitungen 4...7), und so weiter (siehe auch Tabelle 2).

Konfig.-Reg.-Bit:	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
entspricht Hex.-Wert:	80h	40h	20h	10h	08h	04h	02h	01h
entspricht Bin.-Wert:	10000000b	01000000b	00100000b	00010000b	00001000b	00000100b	00000010b	00000001b
I/O-Leitungen:	**31...28**	**27...24**	**23...20**	**19...16**	**15...12**	**11...8**	**7...4**	**3...0**

Tabelle 2: Zuweisung von Konfigurationsregisterbit und I/O-Leitungen

Ein gelöschtes Bit (“0“) programmiert die I/O-Leitungen als Eingang.

Ein gesetztes Bit (“1“) programmiert die I/O-Leitungen als Ausgang.

Beispiel: `POKE(0E0h,00001111b)`
In diesem Beispiel werden die I/O-Leitungen wie bei einer Standard-***ADwin***-Karte konfiguriert; d. h. mit dem 12-AIN-Erweiterungsstecker stehen 16 digitale Eingänge und 16 digitale Ausgänge zur Verfügung.

3 Informationen ausgeben und einlesen

Da bis zu 32 digitale Ein- oder Ausgänge programmiert werden können, ist es nicht mehr möglich die I/O-Leitungen mit den ***ADbasic***-Befehlen DIGIN_WORD, DIGOUT_WORD, CLEAR_DIGOUT, SET_DIGOUT und DIGIN anzusprechen.

Ausnahme: Sind die I/O-Leitungen wie bei einer Standard-***ADwin***-Karte konfiguriert, so kann auf das LOW-Word mit den ***ADbasic***-Befehlen lesend und schreibend zugegriffen werden.

Mit PEEK- und POKE-Befehlen können Informationen aus den Registern ausgelesen bzw. hineingeschrieben werden. Dazu stehen vier 16-Bit breite Register zur Verfügung, über die eine Ein- und Ausgabe abgewickelt werden kann (siehe Tabelle 3).

Hinweis: Beachten Sie bitte, daß Sie bei der Verwendung eines ADSP21062 zu den in Tabelle 3 angegebenen Registeradressen einen Adreßoffset von 204 000 00H addieren müssen. Vergleichen Sie hierzu auch die Angaben in dem ***ADwin***-Hardware-Handbuch.

Register:	**Konfigurieren**	**Einlesen**		**Ausgeben**	
		HIGH-Word	LOW-Word	HIGH-Word	LOW-Word
		Bits: 31...16	Bits: 15...0	Bits: 31...16	Bits: 15...0
Adresse:	**0E0h**	**0B0h**	**1B0h**	**1C0h**	**0C0h**

Tabelle 3: Zuweisung der Registeradressen bei den Prozessoren T400, T450 und T805

3.1 Daten einlesen

Will man die Registerinhalte einlesen, und damit die als Eingänge programmierten Pins abfragen, so kommt der PEEK-Befehl zur Anwendung.

Beispiel:
```
PAR_1 = PEEK(0B0h) 'HIGH-Word einlesen
PAR_2 = PEEK(1B0h) 'LOW-Word einlesen
```

! Achtung: Sind innerhalb eines Wortes (16-Bit) ein oder mehrere I/O-Nibbles als Ausgänge programmiert, so wird der ins Ausgabe-Register geschriebene Wert an diesen Bitpositionen zurückgelesen (auch bekannt unter: REGISTER-READ-BACK).
Sind diese Informationen für die Weiterverarbeitung nicht nötig oder gar störend, so sollten Sie sie durch eine UND-Verknüpfung ausmaskieren.

Beispiel: Um z.B. das höchstwertige Nibble im LOW-Word auszumaskieren, weil dieses Nibble als Ausgang programmiert ist, dann können Sie die folgende Zeile ausführen lassen:

```
PAR_1 = PEEK(1B0h) AND 0FFFh
```

3.2 Daten ausgeben

Sollen die Registerinhalte an den als Ausgänge programmierten Pins ausgegeben werden, so kommt der POKE-Befehl zur Anwendung.

Beispiel:

```
POKE(1C0h,0F81Fh)   'HIGH-W.: 1111 1000 0001 1111 b
POKE(0C0h,7E0h)     'LOW-W. : 0000 0111 1110 0000 b
```

Hinweis: Sind innerhalb eines Wortes (16-Bit) ein oder mehrere I/O-Nibbles als Eingänge programmiert, so hat dies keinen Einfluß bei der Ausgabe der Registerinhalte, d. h. an diesen I/O-Leitungen werden keine Informationen ausgegeben.

4 Beispiel und Test-Programm

Damit das in dieser Dokumentation beschriebene Test-Programm in der gedachten Weise funktioniert, müssen Verbindungen an der Sub-D-Buchse der ***ADwin***-Karte mit ***ADwin-32-I/O***-Option und dem Erweiterungsstecker AIN-12 vorgenommen werden. Diese Verbindungen sind notwendig weil in dem Test-Programm mehrere Leitungen als Ausgänge programmiert sind und von den als Eingängen programmierten Leitungen zurückgelesen werden.

Hinweis: Beachten Sie bitte, daß Sie bei der Verwendung eines ADSP21062 in dem hier abgedruckten Test-Programm zu den Registeradressen einen Adreßoffset von 204 000 00H addieren müssen. Vergleichen Sie hierzu auch die Angaben in dem ***ADwin***-Hardware-Handbuch.

Hinweis: In der folgenden Tabelle finden Sie nur die Pin-Bezeichnungen. Die Zuordnung zwischen den Bezeichnungen der I/O-Leitungen und den Pins entnehmen Sie bitte Tabelle 1 in Kapitel 1.2.

Verbindung von:	Pin-Nr.:		auf:	Pin-Nr.:
ADwin-Karte	16	-	Erw.-Stecker	10
ADwin-Karte	29	-	Erw.-Stecker	20
ADwin-Karte	30	-	***ADwin***-Karte	11
ADwin-Karte	31	-	***ADwin***-Karte	12
ADwin-Karte	32	-	***ADwin***-Karte	13
ADwin-Karte	33	-	***ADwin***-Karte	14
ADwin-Karte	34	-	***ADwin***-Karte	15
Erw.-Stecker	9	-	Erw.-Stecker	29
Erw.-Stecker	8	-	Erw.-Stecker	28
Erw.-Stecker	7	-	Erw.-Stecker	27
Erw.-Stecker	6	-	Erw.-Stecker	26
Erw.-Stecker	5	-	Erw.-Stecker	25
Erw.-Stecker	4	-	Erw.-Stecker	24
Erw.-Stecker	3	-	Erw.-Stecker	23
Erw.-Stecker	2	-	Erw.-Stecker	22
Erw.-Stecker	1	-	Erw.-Stecker	21

Tabelle 4: Verbindungsliste der Teststecker

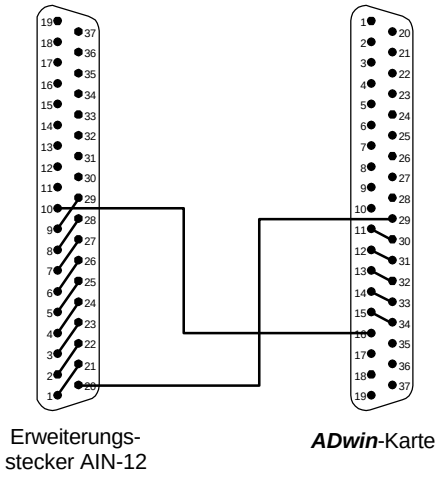


Abbildung 3: Verbindungen der Teststecker

```
'####################################################################
'#                                                                  #
'#                      T E S T - P R O G R A M                     #
'#                                 for                              #
'#           ADwin-card with 32 programmable digital I/O-lines      #
'#                                                                  #
'####################################################################

DIM conf_reg AS INTEGER         'address for configuration register
DIM rd_lw, rd_hw AS INTEGER     'address for READ
DIM wr_lw, wr_hw AS INTEGER     'address for WRITE
DIM dir AS INTEGER              'direction of dataflow
DIM in, out AS INTEGER          'input- and output-word
DIM bit AS INTEGER              'value of pre-shifted bit

'####################################################################
INIT:

  conf_reg=0E0h             'address of configuration register
  rd_lw=0B0h                'read LOW-word
  rd_hw=1B0h                'read HIGH-word
  wr_lw=1C0h                'write LOW-word
  wr_hw=0C0h                'write HIGH-word

  POKE(conf_reg,0Fh)        'IN (0-15) & OUT (16-31) like ADwin-card
  dir=0                     'from DIGOUT to -IN like ADwin-card
  out=1                     'initial output value

'####################################################################
EVENT:

  IF (dir=0) THEN           'DIGOUT to -IN like standard ADwin-card
    POKE(wr_hw,out)         'set output-lines
    in=PEEK(rd_lw)          'read input-lines
    bit=in AND 8000h        'keep value of MSB in mind
    in=SHIFT_LEFT(in,1)     'shift input-word one bit left
    in=in AND 0FFFFh        'only 16 bit of interest
    IF (bit<>0) THEN        'most significant bit set?
      in=in OR 1                 'set least significant bit
    ENDIF
    PAR_1=out               'display output-word
    PAR_2=in                'display input-word
    INC PAR_9               'count of performed loops
    IF (out<>in) THEN       'input unequal output?
      INC PAR_3             'increment error-counter
    ENDIF
    out=SHIFT_LEFT(out,1)   'shift output-word one bit left
    IF (out >0FFFFh) THEN   'set bit was shifted to last position
      out=1                 'reset initial output value
      dir=1                 'swap direction
      POKE(conf_reg,0F0h)   'IN (line16-31) & OUT (line0-15)
    ENDIF
  ENDIF
```

```
  IF (dir=1) THEN          'reverse OUT to IN of standard ADwin-card
    POKE(wr_lw,out)        'set output-lines
    in=PEEK(rd_hw)         'read input-lines
    bit=in AND 1h          'keep value of LSB in mind
    in=SHIFT_RIGHT(in,1)   'shift word one bit right
    in=in AND 0FFFFh       'only 16 bit of interest
    IF (bit<>0) THEN       'MSB was set?
      in=in OR 8000h       'set MSB
    ENDIF
    PAR_1=out              'display output-word
    PAR_2=in               'display input-word
    INC PAR_10             'count of performed loops
    IF (out<>in) THEN      'input equal output?
      INC PAR_3            'increment error-counter
    ENDIF
    out=SHIFT_LEFT(out,1)       'shift output-word one bit left
    IF (out >0FFFFh) THEN       'set bit was shifted to last pos.
      out=1                     'reset initial output value
      dir=0                     'swap direction
      POKE(conf_reg,0Fh)   'IN (0-15) & OUT (16-31)
    ENDIF
  ENDIF

'####################################################################
FINISH:

POKE(conf_reg,00h)         'all I/O-lines as inputs

'####################################################################
```