7 Ein-/Ausgabeanweisungen

7.1 Allgemeines

Die KR C1 kennt 1026 Eingänge und 1024 Ausgänge. Im KUKA-Standardsteuerschrank stehen dem Anwender am Stecker X11 (MFC-Baugruppe) folgende Eingänge und Ausgänge zur Verfügung:

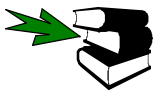
Eingänge	1 ¼ 16	
Ausgänge	1 ¼ 16	(mit max. 100 mA belastbar; 100% Gleichzeitigkeit)
Ausgänge	17 ¼ 20	(mit max. 2 A belastbar; 100% Gleichzeitigkeit)

Optional können weitere Ein-/Ausgänge z.B. über Feldbusse projektiert werden.

Eingänge können gelesen, Ausgänge gelesen und geschrieben werden. Sie werden über die Systemvariablen `$IN[Nr]` bzw. `$OUT[Nr]` angesprochen. Nicht benutzte Ausgänge können als Merker benutzt werden.

Die Ein-/Ausgänge der MFC-Baugruppe können in der Datei "`IO_INF.INI`"auf andere Bereiche umrangiert werden.

Aus Sicherheitsgründen lösen alle Ein-/Ausgabeanweisungen und Zugriffe auf Systemvariablen zur Ein-/Ausgabe einen Vorlaufstop aus.



Abschnitt 4.4

NOTIZEN:

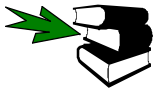
7.2 Binäre Ein-/Ausgänge

Werden Ein-/Ausgänge einzeln angesprochen, so spricht man von binären Ein-/Ausgängen. Binäre Ausgänge können nur 2 Zustände haben: Low oder High. Deshalb werden sie als Variablen vom Typ BOOL behandelt.

Ausgänge können demnach mit den Systemvariablen

```
$OUT[Nr] = TRUE          gesetzt werden, und mit
$OUT[Nr] = FALSE         zurückgesetzt werden.
```

Der Zustand eines Eingangs $IN[Nr] kann in eine boolsche Variable eingelesen werden oder als boolscher Ausdruck in Programmablauf-, Interrupt- oder Triggeranweisungen verwendet werden.



Abschnitt 6 Programmablaufkontrolle,
Abschnitt 9 Interrupt-Behandlung,
Abschnitt 10 Bahnbezogene Schaltaktionen

Die Anweisungsfolgen

```
BOOL SCHALTER
?
SCHALTER = $IN[6]
IF SCHALTER == FALSE THEN
?
ENDIF
```

und

```
IF $IN[6] == FALSE THEN
?
ENDIF
```

haben somit die gleiche Bedeutung.

SIGNAL

In der KR C1 ist es weiterhin möglich, einzelnen Ein- oder Ausgängen Namen zuzuweisen. Hierzu dient die Signalvereinbarung. Sie muß wie alle Vereinbarungen im Vereinbarungsteil des Programms stehen. Man kann also auch

```
SIGNAL SCHALTER $IN[6]
?
IF SCHALTER == FALSE THEN
?
ENDIF
```

programmieren. Die Variable Schalter wird intern wieder als BOOL deklariert.

Systemeingänge und -ausgänge können ebenfalls mit $IN und $OUT angesprochen werden. Systemausgänge sind allerdings schreibgeschützt. Eingang 1025 ist immer TRUE, Eingang 1026 ist immer FALSE. Diese Eingänge werden z.B. in den Maschinendaten als "Dummy"-Variablen benutzt. Eine mehrfache Nutzung ist zulässig.

Wie Ein-/Ausgänge eingesetzt werden, erläutert Beispiel 6.1:

```
DEF  BINSIG ( )
;--------- Deklarationsteil ---------
EXT  BAS (BAS_COMMAND:IN,REAL:IN )
DECL AXIS HOME
SIGNAL ABBRUCH $IN[16]
SIGNAL LINKS $OUT[13]
SIGNAL MITTE $OUT[14]
SIGNAL RECHTS $OUT[15]
;---------- Initialisierung ---------
BAS (#INITMOV,0 ) ;Initialisierung von Geschwindigkeiten,
                  ;Beschleunigungen, $BASE, $TOOL, etc.
HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}

;------------- Hauptteil ------------
PTP  HOME                  ;SAK-Fahrt
LINKS=FALSE
MITTE=TRUE                 ;in mittlerer Stellung
RECHTS=FALSE

WHILE ABBRUCH==FALSE       ;Abbruch, wenn Eingang 16 gesetzt
      IF $IN[1] AND NOT LINKS THEN        ;Eingang 1 gesetzt
      PTP {A1 45}
      LINKS=TRUE                        ;in linker Stellung
      MITTE=FALSE
      RECHTS=FALSE
   ELSE
      IF $IN[2] AND NOT MITTE THEN      ;Eingang 2 gesetzt
         PTP {A1 0}
         LINKS=FALSE
         MITTE=TRUE                      ;in mittlerer Stellung
         RECHTS=FALSE
      ELSE
         IF $IN[3] AND NOT RECHTS THEN  ;Eingang 3 gesetzt
            PTP {A1 -45}
            LINKS=FALSE
            MITTE=FALSE
            RECHTS=TRUE                  ;in rechter Stellung
         ENDIF
      ENDIF
  ENDIF
ENDWHILE
PTP  HOME
END
```

Durch das Setzen der Eingänge 1,2 oder 3 kann der Roboter in drei verschiedene Stellungen verfahren werden. Ist der Roboter in der gewünschten Position angekommen, so wird dies durch Setzen der entsprechenden Ausgänge 13, 14 oder 15 angezeigt.
Da diese Ausgänge also immer die aktuelle Stellung des Roboters anzeigen, kann mit der Abfrage

```
IF $IN[1] AND NOT $OUT[13] THEN
?
ENDIF
```

auch verhindert werden, daß der Roboter bei jedem Durchlauf der While-Schleife erneut versucht, auf die bereits eingenomme Position zu verfahren. Der Roboter verfährt also nur, wenn ein Eingang gesetzt ist (d.h. Anweisung zum Verfahren in gewünschte Position) **und** der zugehörige Ausgang **nicht** gesetzt ist (d.h. Roboter ist noch nicht in dieser Position) (siehe Tab. 23).

Durch Setzen des Eingangs 16 wird die While-Schleife und somit das Programm beendet.

$IN [Nr] AND NOT $OUT[Nr]		$OUT[Nr]	
		TRUE	FALSE
$IN[Nr]	TRUE	FALSE	TRUE
	FALSE	FALSE	FALSE

Tab. 23 Wahrheitstabelle für eine “AND NOT” – Verknüpfung

NOTIZEN:

7.3 Digitale Ein-/Ausgänge

Mit der Signalvereinbarung kann man nicht nur einzelne Ein-/Ausgänge mit Namen versehen, sondern auch mehrere binäre Ein- oder Ausgänge zu einem digitalen Signal zusammenfassen. Mit der Vereinbarung

```
SIGNAL AUS $OUT[10] TO $OUT[20]
```

können nun z.B. die Ausgänge 10 bis 20 über die intern als **Integer** deklarierte Variable AUS als 11-Bit-Wort angesprochen werden.

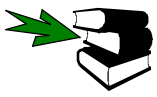
Den so deklarierten Digitalausgang kann man über jede mögliche Integer-Zuweisung an die Variable AUS beschreiben, z.B.:

```
AUS = 35
AUS = 'B100011'
AUS = 'H23'
```

- **G** Ein-/Ausgänge müssen in der Signalvereinbarung lückenlos und in aufsteigender Reihenfolge angegeben werden.
- **G** Es können höchstens 32 Ein- bzw. Ausgänge zu einem Digitalsignal zusammengefaßt werden.
- **G** Ein Ausgang darf in mehreren Signalvereinbarungen vorkommen.

Wenn man die Ausgänge 13 bis 15 von Beispiel 6.1 damit zu einer Variablen POSITION zusammenfaßt, ergibt sich folgendes modifizierte Programm:

```
DEF  BINSIG_D ( )

;--------- Deklarationsteil ---------
EXT  BAS (BAS_COMMAND:IN,REAL:IN )
DECL AXIS HOME
SIGNAL ABBRUCH $IN[16]
SIGNAL POSITION $OUT[13] TO $OUT[15]

;---------- Initialisierung ---------
BAS (#INITMOV,0 ) ;Initialisierung von Geschwindigkeiten,
                  ;Beschleunigungen, $BASE, $TOOL, etc.
HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}

;------------- Hauptteil ------------
PTP  HOME                  ;SAK-Fahrt

POSITION='B010'            ;in mittlerer Stellung

WHILE ABBRUCH==FALSE       ;Abbruch, wenn Eingang 16 gesetzt

   IF $IN[1] AND (POSITION<>'B001') THEN      ;Eingang 1 gesetzt
      PTP {A1 45}
      POSITION='B001'                        ;in linker Stellung
   ELSE
      IF $IN[2] AND (POSITION<>'B010') THEN   ;Eingang 2 gesetzt
         PTP {A1 0}
         POSITION='B010'                  ;in mittlerer Stellung
      ELSE
         IF $IN[3] AND (POSITION<>'B100') THEN;Eingang 3 gesetzt
            PTP {A1 -45}
            POSITION='B100'                ;in rechter Stellung
         ENDIF
      ENDIF
   ENDIF

ENDWHILE

PTP  HOME
END
```



Abschnitt 7.6

NOTIZEN:

7.4 Impulsausgänge

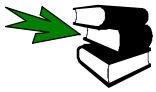
PULSE

Mit der PULSE-Anweisung können einzelne Ausgänge für eine bestimmte Dauer gesetzt oder rückgesetzt werden. Die Anweisung

```
PULSE($OUT[4],TRUE,0.7)
```

setzt beispielsweise Ausgang 4 für eine Zeit von 0.7 Sekunden auf High-Pegel. Der Impuls kann dabei parallel zum Roboterprogramm ablaufen (der Interpreter wird dabei nicht angehalten).

Anstatt der direkten Angabe des Ausgangs mit $OUT[*Nr*] kann auch eine Signalvariable stehen.



Abschnitt 7.2

Die realisierbaren Impulszeiten liegen zwischen 0.1 und 3.0 Sekunden. Das Raster beträgt 0.1 Sekunden. Die Steuerung rundet alle Werte auf ein Zehntel.

- **Es dürfen maximal 16 Impulsausgänge gleichzeitig programmiert werden.**
- **Zu lange Impulszeiten werden erst während des Programmlaufs erkannt und lösen einen Programmstop aus.**
- **Es können sowohl High-Impulse, als auch Low-Impulse programmiert werden.**
- **Bei "Programm RESET" oder "Programm abwählen" wird der Impuls abgebrochen.**
- **Ein anstehender Impuls kann durch Interrupts beeinflußt werden.**
- **Impulsausgänge können auch in der Steuerungsebene programmiert werden.**
- **Die PULSE-Anweisung löst einen Vorlaufstop aus. Nur in der TRIGGER-Anweisung wird sie bewegungsbegleitend ausgeführt.**

Ein Impuls wird NICHT abgebrochen bei

- **NOT-AUS, Bedienstop oder Fehlerstop,**
- **Erreichen des Programmendes (END-Anweisung),**
- **Loslassen der Starttaste, wenn der Impuls vor der ersten Bewegungsanweisung programmiert wurde, und der Roboter SAK noch nicht erreicht hat.**

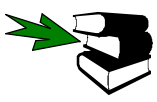
Im nächsten Programm finden Sie einige Beispiele zur Anwendung der PULSE-Anweisung:

```
DEF  PULSIG ( )

;--------- Deklarationsteil ---------
EXT  BAS (BAS_COMMAND  :IN,REAL  :IN )
DECL AXIS HOME
INT I
SIGNAL OTTO $OUT[13]

;---------- Initialisierung ---------
BAS (#INITMOV,0 ) ;Initialisierung von Geschwindigkeiten,
                  ;Beschleunigungen, $BASE, $TOOL, etc.
HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}

FOR I=1 TO  16
$OUT[I]=FALSE     ;alle Ausgaenge auf LOW setzen
ENDFOR

;------------- Hauptteil ------------
PULSE ($OUT[1],TRUE,2.1)  ;Impuls kommt direkt fuer 2.1s
PTP  HOME                 ;SAK-Fahrt

OTTO=TRUE                 ;Ausgang 13 auf TRUE setzen
PTP  {A3 45,A5 30}
PULSE (OTTO,FALSE,1.7)    ;LOW-Impuls fuer 1.7s auf Ausgang 13
                          ;Impuls kommt erst nach Bewegung
WAIT SEC 2

FOR I=1 TO  4
PULSE ($OUT[I],TRUE,1)    ;nacheinander die Ausgaenge 1-4
WAIT SEC 1                ;fuer 1s auf High
ENDFOR

;bahnbezogenes Aufbringen eines Impulses
TRIGGER  WHEN DISTANCE=0 DELAY=50 DO PULSE ($OUT[8],TRUE,1.8)
LIN  {X 1391,Y -319,Z 1138,A -33,B -28,C -157}

PTP  HOME
CONTINUE                 ;Vorlaufstop verhindern fuer Ausgang 15
PULSE ($OUT[15],TRUE,3)   ;Impuls kommt direkt (im
 Vorlauf)
                          ;auf Ausgang 15
PULSE ($OUT[16],TRUE,3)   ;Impuls kommt erst nach HOME-Fahrt
END                       ;und steht noch nach END an
```

Beachten Sie in diesen Beispielen genau, ab wann die programmierten Impulse an den Ausgängen anliegen: Grundsätzlich führt die PULSE-Anweisung immer zu einem Stop des Rechnervorlaufs. Der Impuls liegt also erst nach Bewegungsbeendigung an.

Es gibt zwei Wege den Vorlaufstop zu verhindern:

- Programmierung einer CONTINUE-Anweisung direkt vor der PULSE-Anweisung
- Verwendung der PULSE-Anweisung in einer TRIGGER-Anweisung (bahnbezogene Schaltaktion)



Abschnitt 4.4 (CONTINUE) und Abschnitt 10 (TRIGGER)

7.5 Analoge Ein-/Ausgänge

Neben den binären Ein-/Ausgängen kennt die KR C1 auch analoge Ein-/Ausgänge. Über optionale Bussysteme stellt die KR C1 8 analoge Eingänge und 16 analoge Ausgänge zur Verfügung. Ausgänge können mit den Systemvariablen $ANOUT[1] ¼ $ANOUT[16] gelesen oder geschrieben werden, Eingänge können mit den Variablen $ANIN[1] ¼ $ANIN[8] nur gelesen werden.

ANIN
ANOUT

Analoge Ein- und Ausgänge können sowohl statisch als auch dynamisch, d.h. durch ständige Abfrage im Interpolationstakt (z.Z 12 ms), angesprochen werden. Während das statische Lesen und Schreiben wie bei den binären Signalen durch einfache Wertzuweisungen erfolgt, benutzt man zum zyklischen Bearbeiten die speziellen Anweisungen ANIN und ANOUT.

7.5.1 Analoge Ausgänge

Die Ausgabewerte für die 16 analogen Ausgänge der KR C1 liegen zwischen -1.0 ¼ +1.0 und sind normiert auf die Ausgangsspannung ±10.0 V. Falls der Ausgabewert die Grenzen ±1.0 überschreitet wird der Wert abgeschnitten.

Zum Setzen eines Analogkanals weist man der entsprechenden $ANOUT-Variable einfach einen Wert zu:

```
$ANOUT[2] = 0.5              ;Analogkanal 2 wird auf +5V gesetzt
```

oder

```
REAL V_KLEBER
?
V_KLEBER = -0.9
$ANOUT[15] = V_KLEBER        ;Analogkanal 15 wird auf -9V gesetzt
```

Diese Zuweisungen sind statisch, da der Wert des beaufschlagten Kanals sich erst ändert, wenn der betreffenden Systemvariablen $ANOUT[*Nr*] explizit ein neuer Wert zugewiesen wird.

Oft ist es jedoch wünschenswert, daß ein bestimmter analoger Ausgang während der Programmabarbeitung ständig in einer festgelegten Zykluszeit neu berechnet wird. Diese dynamische Analogausgabe erfolgt mit der ANOUT-Anweisung. Mit der Anweisung

```
ANOUT ON DRAHT = 0.8 * V_DRAHT
```

können Sie z.B. durch einfache Wertzuweisung an die Variable V_DRAHT den mit der Signalvariablen DRAHT spezifizierten Analogausgang verändern. Die Spannung am entsprechenden Ausgang folgt somit der Variablen V_DRAHT.
Die Variable DRAHT muß vorher natürlich mit der SIGNAL-Vereinbarung deklariert werden, z.B.:

```
SIGNAL DRAHT $ANOUT[2]
```

Mit

```
ANOUT OFF DRAHT
```

wird die zyklische Analogausgabe wieder beendet.

Der zyklisch aktualisierte Ausdruck, der für die Berechnung des Analog-Ausgabewertes angegeben werden muß, darf eine gewisse Komplexität nicht überschreiten. Die zulässige Syntax ist deshalb eingeschränkt und an der Technologie orientiert. Die vollständige Syntax lautet

ANOUT ON

```
ANOUT ON Signalname = Faktor * Regelglied á+ Offsetñ áDELAY=Zeitñ
```

für das Starten der zyklischen Analogausgabe, bzw.

ANOUT OFF

```
ANOUT OFF Signalname
```

für das Beenden. Die Bedeutung der einzelnen Argumente ist aus Tab. 24 ersichtlich.

Argument	Datentyp	Bedeutung
Signalname	REAL	Signalvariable, die den analogen Ausgang spezifiziert (muß mit SIGNAL deklariert sein). Nicht zulässig ist eine direkte Angabe von $ANOUT[*Nr*].
Faktor	REAL	beliebiger Faktor, der eine Variable, ein Signalname oder eine Konstante sein kann
Regelglied	REAL	Über das Regelglied wird der Analogausgang beeinflußt. Es kann eine Variable oder ein Signalname sein.
Offset	REAL	Optional kann ein Offset zum Regelglied programmiert werden. Der Offset muß eine Konstante sein.
Zeit	REAL	Mit dem Schlüsselwort DELAY und einer positiven oder negativen Zeitangabe in Sekunden kann optional das zyklisch berechnete Ausgangssignal verzögert (+) oder vorzeitig (–) ausgegeben werden.

Tab. 24 Argumente in der ANOUT-Anweisung

NOTIZEN:

7.5.2 Analoge Eingänge

Die 8 analogen Eingänge der KR C1 können über die Variablen $ANIN[1] bis $ANIN[8] durch einfache Wertzuweisung an eine REAL–Variable gelesen werden:

```
REAL TEIL
?
TEIL = $ANIN[3]
```

oder

```
SIGNAL SENSOR3 $ANIN[3]
REAL TEIL
?
TEIL = SENSOR3
```

Die Werte in $ANIN[*Nr*] bewegen sich zwischen +1.0 und –1.0 und repräsentieren eine Eingangsspannung von +10V bis –10V.

ANIN

Zum zyklischen Lesen von Analogeingängen dient die ANIN–Anweisung. Mit ihr können bis zu 3 analoge Eingänge gleichzeitig eingelesen werden. Das Lesen erfolgt dabei im Interpolationstakt.

Mit der Anweisungsfolge

```
SIGNAL SENSOR3 $ANIN[3]
REAL TEIL
?
ANIN ON TEIL = 1 * SENSOR3
```

können Sie somit den analogen Eingang 3 zyklisch lesen, und mit der der Anweisung

```
ANIN OFF SENSOR3
```

das Lesen wieder beenden.

Zu beachten ist, daß zur gleichen Zeit höchsten 2 ANIN ON – Anweisungen aktiv sein dürfen. Es ist zulässig in beiden Anweisungen auf die selbe Analogschnittstelle zuzugreifen, oder die gleiche Variable zu beschreiben.

Die vollständige Syntax zum zyklische Lesen eines Analogeinganges lautet:

ANIN ON

```
ANIN ON Wert = Faktor * Signalname á Offsetñ
```

Das Beenden der zyklischen Überwachung wird mit

ANIN OFF

```
ANIN OFF Signalname
```

eingeleitet. Die Bedeutung der Argumente entnehmen Sie Tab. 25.

Argument	Datentyp	Bedeutung
Wert	REAL	Der Wert kann eine Variable oder ein (Ausgangs–)Signalname sein. In Wert wird das Ergebnis des zyklischen Lesens abgelegt.
Signalname	REAL	Signalvariable, die den analogen Eingang spezifiziert (muß mit SIGNAL deklariert sein). Nicht zulässig ist eine direkte Angabe von $ANIN[*Nr*].
Faktor	REAL	beliebiger Faktor, der eine Variable, ein Signalname oder eine Konstante sein kann
Offset	REAL	Optional kann ein Offset programmiert werden. Der Offset kann eine Konstante, eine Variable oder ein Signalname sein.

Tab. 25 Argumente in der ANIN–Anweisung

Im nachfolgenden Beispiel sind die Anweisungen zur Analogein- und ausgabe illustriert. Mit Hilfe der Systemvariablen $TECHIN[1] und einem an einem Analogeingang angeschlossenen Bahnfolgesensor kann so z.B. eine Bahnkorrektur während der Bewegung vorgenommen werden. Die Variable $VEL_ACT, die stets die aktuelle Bahngeschwindigkeit enthält, kann mit entsprechenden Faktoren gewichtet zu einer geschwindigkeitsproportionalen Analogausgabe benutzt werden, also z.B. um die Klebermenge beim Kleberauftrag zu steuern.


```
DEF  ANSIG ( )

;--------- Deklarationsteil ---------
EXT  BAS (BAS_COMMAND  :IN,REAL  :IN )
DECL AXIS HOME
INT I
SIGNAL KLEBER $ANOUT[1]       ;Duesenoeffnung fuer Kleber
SIGNAL KORREKTUR $ANIN[5]     ;Bahnfolgesensor

;---------- Initialisierung ---------
BAS (#INITMOV,0 ) ;Initialisierung von Geschwindigkeiten,
                  ;Beschleunigungen, $BASE, $TOOL, etc.
HOME={AXIS: A1 0,A2 -90,A3 90,A4 0,A5 0,A6 0}

FOR I=1 TO  16
$ANOUT[I]=0       ;alle Ausgaenge auf 0V setzen
ENDFOR

;------------- Hauptteil ------------
PTP  HOME                  ;SAK-Fahrt

$ANOUT[3] = 0.7            ;Analogausgang 3 auf 7V

IF $ANIN[1] >= 0 THEN      ;Klebevorgang nur, wenn Analogeingang
1
                           ;positive Spannung hat
   PTP POS1

   ;Bahnkorrektur entsprechend Sensorsignal mit Hilfe der
   ;Systemvariablen $TECHIN
   ANIN ON $TECHIN[1] = 1 * KORREKTUR + 0.1

   ;geschwindigkeitsproportionale Analogausgabe;Systemvariable
$VEL_ACT enthält die aktuelle Bahngeschwindigkeit
   ANOUT ON KLEBER = 0.5 * $VEL_ACT + 0.2 DELAY = -0.12

   LIN POS2
   CIRC HILFSPOS,POS3
   ANOUT OFF KLEBER
   ANIN OFF KORREKTUR

   PTP POS4

ENDIF

PTP HOME
END
```

7.6 Vordefinierte Digitaleingänge

Die Steuerung stellt 6 digitale Eingänge zur Verfügung, welche über die Signalvariablen $DIGIN1¼ $DIGIN6 gelesen werden können. Die Eingänge sind Teil der normalen Anwendereingänge. Sie können eine Länge von 32 Bit und einen dazugehörigen Strobe–Ausgang haben.

Die Projektierung der digitalen Eingänge erfolgt in den Maschinendaten: "/mada/steu/$maschine.dat". Mit einer Signalvereinbarung wird zunächst der Bereich und die Größe des Digitaleinganges festgelegt:

```
SIGNAL $DIGIN3 $IN[1000] TO $IN[1011]
```

Über die weiteren Systemvariablen $DIGIN1CODE¼ $DIGIN6CODE, $STROBE1¼ $STROBE6 und $STROBE1LEV¼ $STROBE6LEV werden die Vorzeicheninterpretationen, die zugehörigen Strobe–Ausgänge und die Art des Strobe–Signals festgelegt:

```
DECL DIGINCODE $DIGIN3CODE = #UNSIGNED ;ohne Vorzeichen
SIGNAL $STROBE3 $OUT[1000]             ;Strobe-Ausgang festlegen
BOOL $STROBE3LEV = TRUE                ;Strobe ist ein High-Impuls
```

Ein Strobe–Ausgang ist ein Ausgang der KR C1 mit einem bestimmten Impuls, der das Signal eines externen Gerätes (z.B. Drehgeber) zum Lesen einfriert.

Während verschiedene Digitaleingänge auf den selben Eingang zugreifen können, dürfen die Strobe–Signale NICHT den selben Ausgang beschreiben.

Der Wertebereich von $DIGIN1¼ $DIGIN6 hängt von der definierten Bit–Länge sowie von der Vorzeicheninterpretation (#SIGNED oder #UNSIGNED) ab:

12 Bit mit Vorzeichen (#SIGNED)	Wertebereich:	–2048¼ 2047
12 Bit ohne Vorzeichen (#UNSIGNED)	Wertebereich:	0¼ 4095

Die Digitaleingänge können entweder statisch durch eine gewöhnliche Wertzuweisung gelesen werden:

```
INT ZAHL
?
ZAHL = $DIGIN2
```

DIGIN

oder aber zyklisch mit einer DIGIN–Anweisung:

```
INT ZAHL
?
DIGIN ON ZAHL = FAKTOR * $DIGIN2 + OFFSET
?
DIGIN OFF $DIGIN2
```

Zur gleichen Zeit sind 2 DIGIN ON – Anweisungen zulässig, wobei beide den gleichen Digitaleingang lesen können. In der DIGIN ON – Anweisung kann auch auf analoge Eingangssignale zugegriffen werden (z.B.. als FAKTOR). Die Syntax ist völlig analog zur ANIN ON–Anweisung:

DIGIN ON

```
DIGIN ON Wert = Faktor * Signalname á  Offsetñ
```

DIGIN OFF

```
DIGIN OFF Signalname
```

Die Bedeutung der einzelnen Argumente ist in Tab. 26 beschrieben.

Argument	Datentyp	Bedeutung
Wert	REAL	Der Wert kann eine Variable oder ein (Ausgangs-)Signalname sein. In Wert wird das Ergebnis des zyklischen Lesens abgelegt.
Signalname	REAL	Signalvariable, die den digitalen Eingang spezifiziert. Zulässig sind nur $DIGIN1¼$DIGIN6
Faktor	REAL	beliebiger Faktor, der eine Variable, ein Signalname oder eine Konstante sein kann
Offset	REAL	Optional kann ein Offset programmiert werden. Der Offset kann eine Konstante, eine Variable oder ein Signalname sein.

Tab. 26 Argumente in der DIGIN-Anweisung

NOTIZEN: