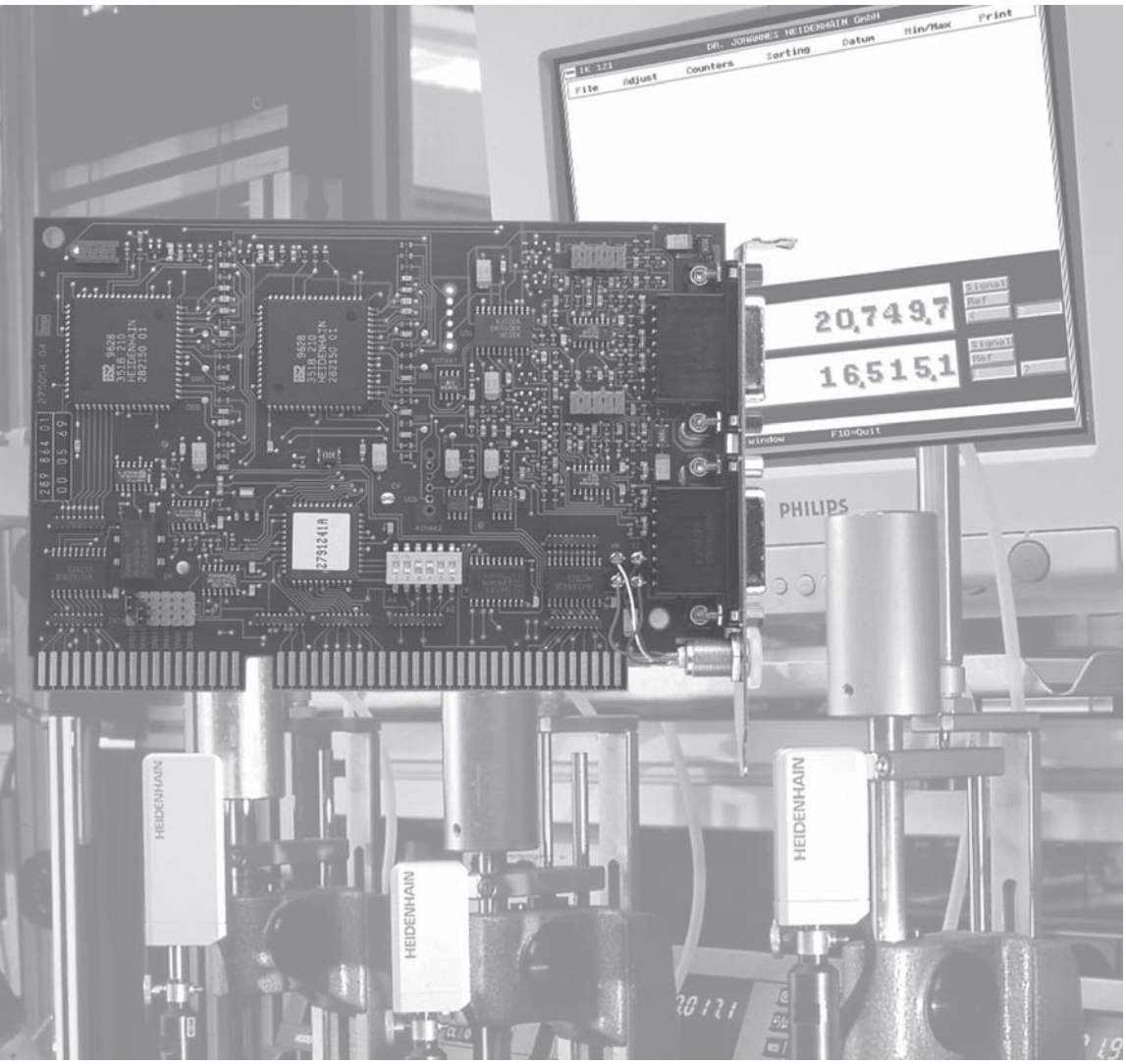


HEIDENHAIN

Manuel d'utilisation

IK 121

Carte de comptage pour PC
et raccordement de systèmes
de mesure HEIDENHAIN

10/98

Sommaire

Contenu de la fourniture

Carte de comptage IK 121 pour PC, exemples de programmation, logiciel driver version et Mode d'emploi

Versions

IK 121 A
290 155-xx

Carte de comptage pour PC (entrées systèmes de mesure avec signaux de courant sinus (1 1 μA_{CC})).

IK 121 V
291 768-xx

Carte de comptage pour PC (entrées systèmes de mesure avec signaux de tension sinus (1 V_{CC})).

Directive de compatibilité électro-magnétique 89/336/EWG

Les dispositions de la directive 89/336/EWG ont été contrôlées à l'aide de l'ordinateur COMPAQ DESKPRO 386/20e.

Accessoires

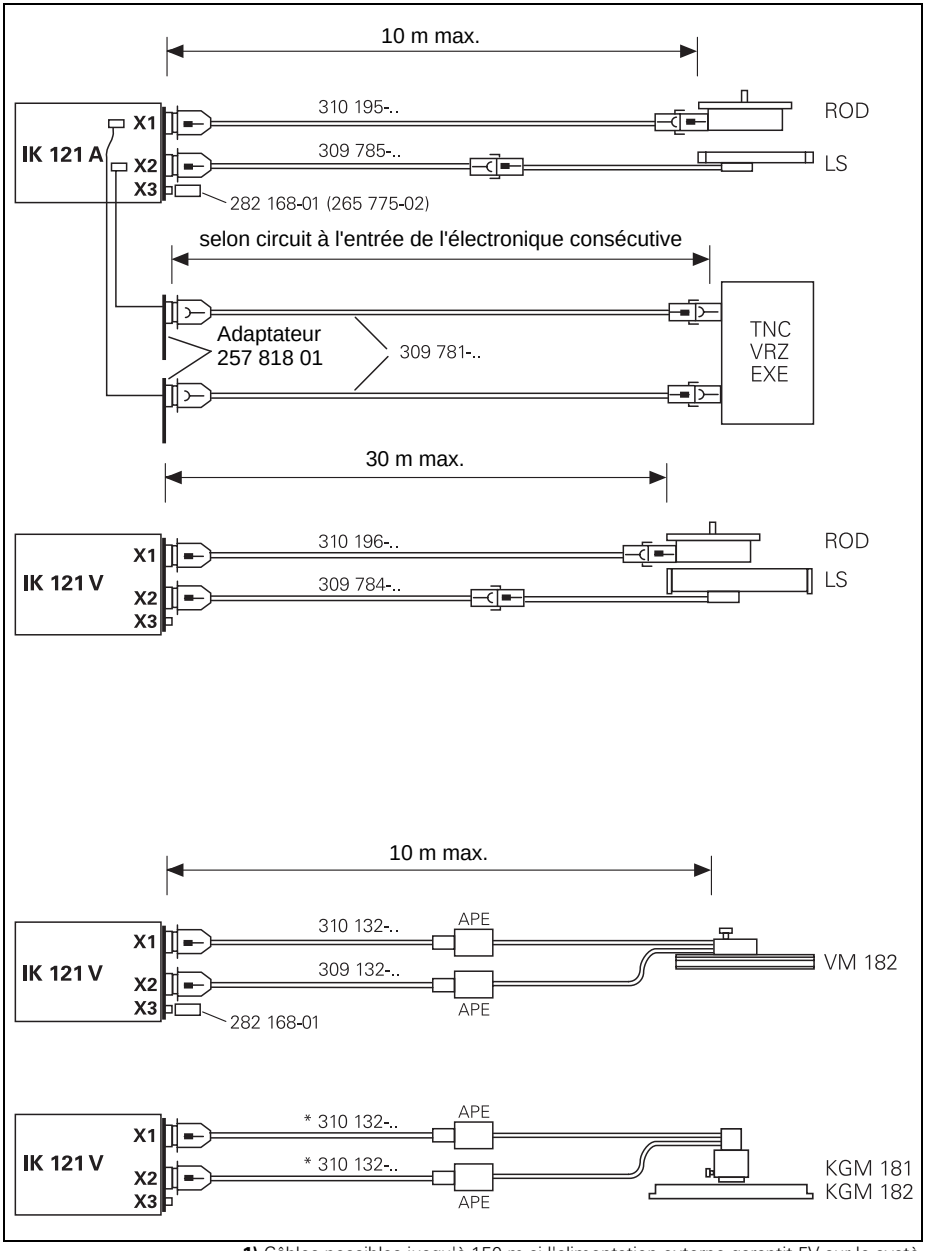
257 818-01	Raccordement Sub-D supplémentaire pour retransmission des signaux du système de mesure des entrées X1 ou X2 vers une autre visualisation ou une autre commande
309 781-xx	Câble de liaison entre le raccordement Sub-D supplémentaire et l'autre visualisation ou l'autre commande
282 168-01 (265 775-02)	Connecteur pour fonctions externes sur le raccordement X3 (deux femelles, deux mâles)

IK 121 A

309 785-xx 9/9 plots	Câble adaptateur avec prise d'accouplement pour systèmes de mesure HEIDENHAIN; Longueur standard 0,5 m
310 195-xx 9/9 plots	Câble adaptateur avec connecteur pour systèmes de mesure HEIDENHAIN équipés d'une embase; Longueur standard 0,5 m

IK 121 V

309 784-xx 15/12 plots	Câble adaptateur avec prise d'accouplement pour systèmes de mesure HEIDENHAIN; Longueur standard 0,5 m
310 196-xx 15/12 plots	Câble adaptateur avec connecteur pour systèmes de mesure HEIDENHAIN équipés d'une embase; Longueur standard 0,5 m



1) Câbles possibles jusqu'à 150 m si l'alimentation externe garantit 5V sur le système de mesure.

Remarques importantes

Danger pour composants internes!
Respecter les consignes de sécurité en matière de manipulation de **composants susceptibles d'être endommagés par des décharges électrostatiques (ESD)** selon DIN EN 100 015. Pour le transport, utiliser exclusivement un conditionnement antistatique. Prévoir une mise à la terre satisfaisante du poste de travail et du technicien.

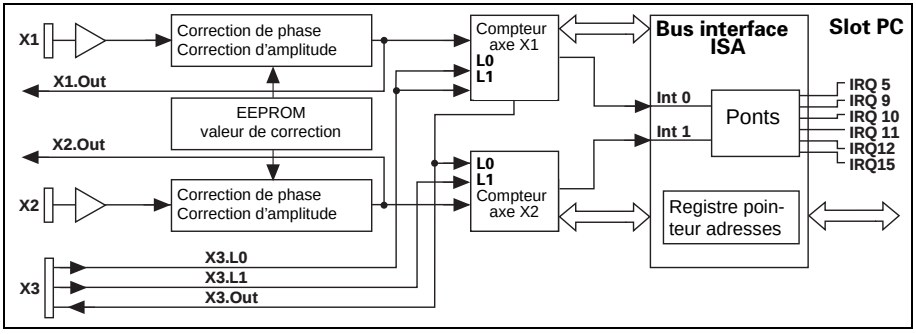
READ.ME

Le fichier READ.ME contient les principales informations relatives à l'installation de l'IK□121 et au logiciel contenu dans la fourniture. Ce fichier est affiché page-à-page à l'écran par le programme README.EXE.

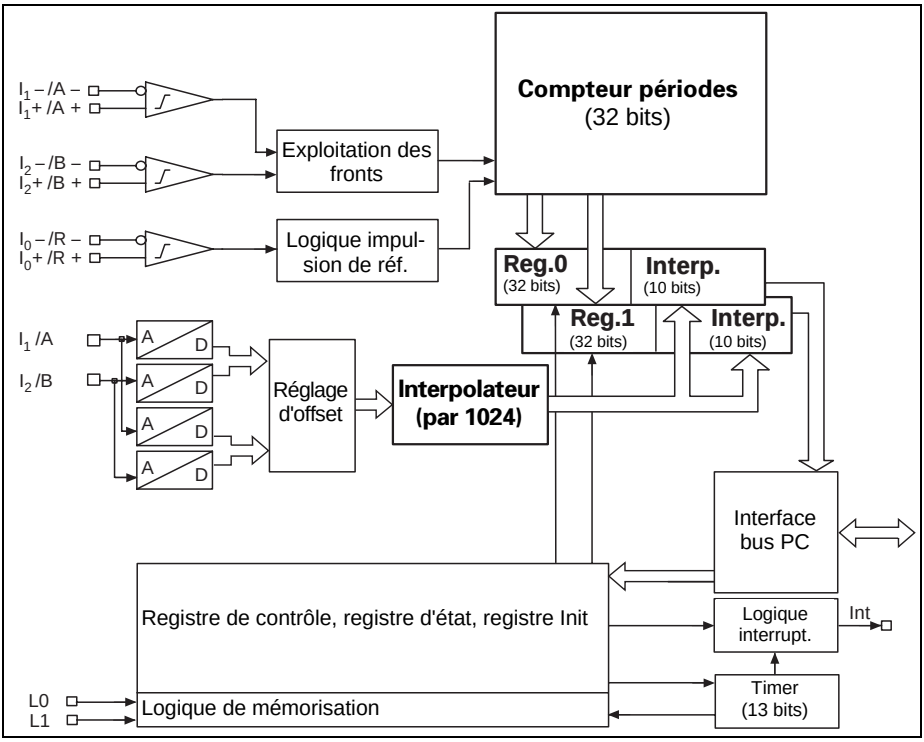
Description technique de l'IK□121

La carte de comptage IK□121 pour PC est insérée directement dans un slot d'extension d'un PC compatible AT. Deux système s de mesure HEIDENHAIN délivrant des signaux sinusoïdaux de courant (IK 121 A) ou de tension (IK 121 V) peuvent être raccor dés. A l'aide du logiciel, les positions des deux systèmes de me sure sont affichées sur votre PC où elles sont mémorisées pour leur traitement ultérieur.
L'IK 121 est idéale pour les applications qui requièrent à la fois une résolution élevée des signaux de mesure et l'acquisition ra pide des valeurs de la mesure.

Synoptique modulaire de l'IK 121



Synoptique modulaire du circuit du compteur



Dans l'IK 121, l'électronique d'interpolation subdivise par 1024 la période du signal d'entrée.
La valeur de mesure 42 bits est constituée par la valeur d'interpolation (10 bits) et la valeur du compteur de périodes (32 bits).
Les valeurs de mesure sont mémorisées dans des registres 48 bits; les bits supérieurs sont élargis en représentation du complément à deux en tenant compte du signe.
Les valeurs de mesure sont appelées et mémorisées soit par les entrées d'appel externes, soit par logiciel ou timers ou bien encore par franchissement des marques de référence avec l'adressage de port.

Les expressions „appel" ou „appeler" utilisées dans ce Manuel signifient que la valeur de comptage dans le registre de données 0 ou dans le registre de données 1 est retenue. Cette valeur de comptage doit être lue par le logiciel et mémorisée dans le PC ou affichée à l'écran.

La phase et l'amplitude des signaux sinusoïdaux délivrés par les systèmes de mesure peuvent être réglés à l'aide du logiciel et de potentiomètres électroniques; le réglage d'offset peut être effectué à l'aide des registres de données à l'intérieur du compteur.

Temps d'accès aux valeurs de mesure

Le temps d'accès aux valeurs de mesure est d'environ 35 µs.

Hardware

Spécification du bus PC

L'IK□121 peut être implantée sur tous les IBM AT et PC compatibles à 100%. HEIDENHAIN ne peut garantir un fonctionnement parfait de l'IK□121 sur des PC qui ne seraient pas compatibles à 100%. L'IK□121 est conforme à la norme internationale IEEE□ P996 définissant le bus AT et ISA (standard industriel).

Largeur du bus	16 bits
Tension d'alimentation	+5 V ±5% +12 V ±5% –12 V ±5%
Consommation en courant	env. 1 watt sans système de mesure
Version de slot	compatible AT, slot court 16 bits

Entrées pour systèmes de mesure sur IK 121 A

L'IK□121 A est prévue pour le raccordement de systèmes de mesure linéaire ou angulaire HEIDENHAIN délivrant des signaux de courant sinusoïdaux I_1 et I_2 ainsi qu'un signal de référence I_0.

Amplitudes du signal I_1, I_2 (0°, 90°) I_0 (marque de référence)	 7 μA_{CC} à 16 μA_{CC} 3,5 µA à 8 µA
Amplitude du signal pour message d'erreur	≤ 2,5 μA_{CC}
Fréquence d'entrée max.	100 kHz
Longueur de câble	10 m max.

Raccordements X1, X2 pour systèmes de mesure
Raccordement Sub-D femelle (9 plots)

Plot n°	Affectation
1	I_1 –
2	0 V (U_N)
3	I_2 –
4	blindage interne
5	I_0 –
6	I_1 +
7	5 V (U_P)
8	I_2 +
9	I_0 +
Boîtier	blindage externe

Entrées pour systèmes de mesure sur IK 121 V

L'IK□121 A est prévue pour le raccordement de systèmes de m esure linéaire ou angulaire HEIDENHAIN délivrant des signaux d e tension sinusoïdaux A et B ainsi qu'un signal de référence R.

Amplitudes du signal A, B (0°, 90°) R (marque de référence)	 0,6 V_{CC} à 1,2 V_{CC} 0,2 V à 0,85 V
Amplitude du signal pour mes sage d'erreur	≤ 0,22 V_{CC}
Fréquence d'entrée max.	400 kHz
Longueur de câble	30 m max.

1) Câbles possibles jusqu'à 150 m si l'alimenation externe garantit 5V sur le systè me de mesure.
Dans ce cas, la fréquence d'entrée est réduite à 250 kHz max.

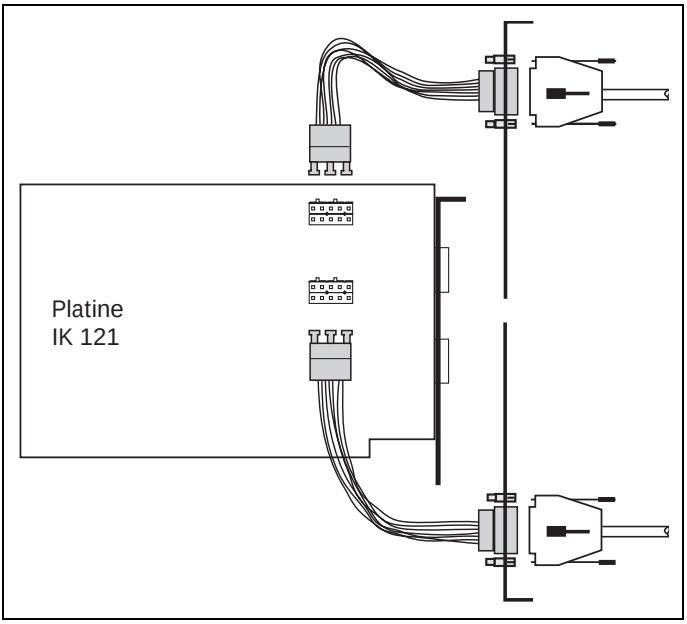
Raccordements X1, X2 pour systèmes de mesure
Raccordement Sub-D femelle (15 plots)

N° du raccordement	Affectation
1	A +
2	0 V (U_N)
3	B +
4	+ 5 V (U_P)
5	ne pas raccorder
6	ne pas raccorder
7	R –
8	ne pas raccorder
9	A –
10	0 V (palpeur)
11	B –
12	+ 5 V (palpeur)
13	ne pas raccorder
14	R +
15	ne pas raccorder
Boîtier	blindage externe

Sorties systèmes de mesure

L'IK 121 restitue également les signaux des systèmes de mesure des entrées X1 et X2 sur deux connecteurs AMP 10□plots sous forme de **signaux de courant sinusoïdaux** (11□μA_{CC}). Ces raccordements peuvent être dirigés vers les raccordements Sub-D 9 plots à l'aide de kits de câble supplémentaires équipés d'un couvercle de slot (Id.-Nr.□257□818□01). Câbles adaptateurs (Id.-Nr.□309□781□..)
pour raccordement sur visualisations de cotes ou électroniques d'interpolation HEIDENHAIN livrables en accessoires
(cf. „Contenu de la fourniture", „Accessoires").

La longueur maximale du câble dépend du circuit d'entrée de l' électronique consécutive.



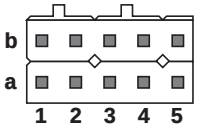
Sorties systèmes de mesure (Id.-Nr. 257 818 01)
Raccordement Sub-D mâle (9 plots)

Plot n°	Affectation
1	I_1 –
2	0 V (U_N)
3	I_2 –
4	non raccordé
5	I_0 –
6	I_1 +
7	non raccordé
8	I_2 +
9	I_0 +
Boîtier	blindage externe

Connecteur de platine pour sorties des systèmes de mesur e
AMP mâle (10 plots)

Plot n° [1)]	Signal
1a	non raccordé
1b	non raccordé
2a	non raccordé
2b	0 V (U_N)
3a	I_0 –
3b	I_0 +
4a	I_2 –
4b	I_2 +
5a	I_1 –
5b	I_1 +

1) La face avec les broches de blocage correspond à b.
Les raccordements 1a et 1 b sont situés sur la face avec encoche.



Alignement des signaux des systèmes de mesure

Les signaux délivrés par les systèmes de mesure peuvent être alignés de la manière suivante:

- Phase et amplitude avec potentiomètres électroniques
- Symétrie (offset) dans les compteurs à l'aide des registres d 'offset

Le contrôle des potentiomètres a lieu par bus I²C. La programm ation des séquences de commande étant complexe, il convient d'utiliser POTIS.EXE ou ADJUST.EXE pour l'alignement des sig naux de mesure.

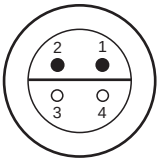
Lorsqu'une application exige le contrôle des potentiomètres, vo us pouvez alors utiliser les fonctions et procédures en „TURBO PASCAL" du fichier IIC.PAS ou bien vous en servir comme base d'orientation.

Les valeurs de correction pour la phase et l'amplitude sont pr otégées en mémorisation dans les composants des potentio mètres électroniques. Les registres d'offset contenus dans le s compteurs ne sont pas protégés en cas de coupure d'alime ntation. Pour cette raison les valeurs de correction d'offset so nt mémorisées dans une EEPROM du composant pour les po tentiomètres électro-
niques. A la mise sous tension, les valeurs de correction d'off set doivent être chargées de l'EEPROM dans les registres d'o ffset des compteurs. Deux procédures destinées à ces tâches sont définies dans le fichier IIC. PAS. La procédure „store_off set" mémorise les valeurs de correction d'offset dans l'EEPR OM. La procédure „load_offset" lit les valeurs de correction d' offset dans l'EEPROM et les copie dans le registre d'offset de s compteurs. „Load_offset" est également utilisé par la procé dure „init_IK121".

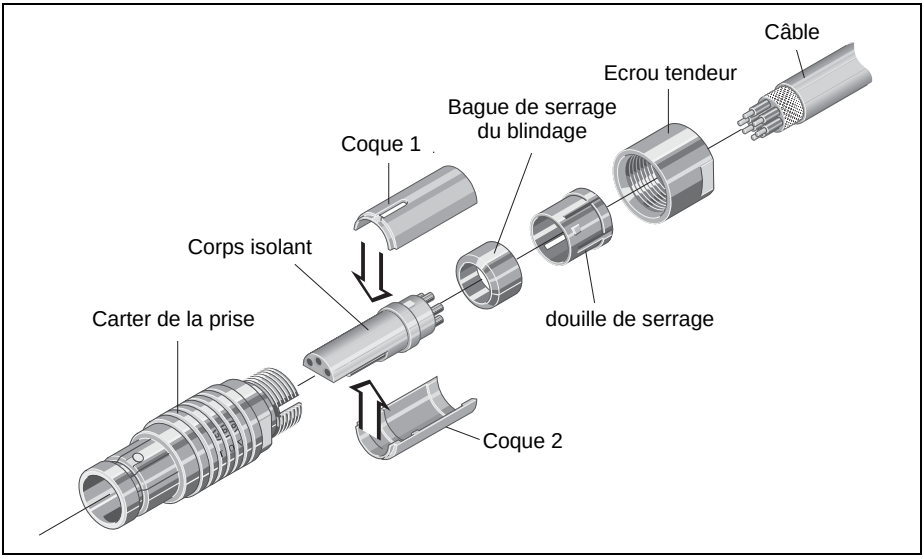
Fonctions externes

Une embase 4 plots sert aux fonctions externes. Le connecteur correspondant (Id.-Nr.□282□168□01) est livrable par HEIDENHA IN.

Montage du connecteur pour les fonctions externes



Vue arrière
o = femelle, • = mâle



Raccordement X3 pour fonctions externes
Embase mâle/femelle (4 plots)

Plot n°	Affectation
1	Entrée: appel valeur de mesure X1 (X3.L0)
2	Entrée: appel valeur de mesure X2 (X3.L1)
3	Sortie: appel valeur de mesure(X3.Out)
4	0 V

Appel des valeurs de mesure sur entrées externes
L'IK□121 est équipée de deux entrées externes sur l'embase X 3 permettant d'appeler et de mémoriser les valeurs de mesure.

Ces entrées peuvent être utilisées pour les interruptions.

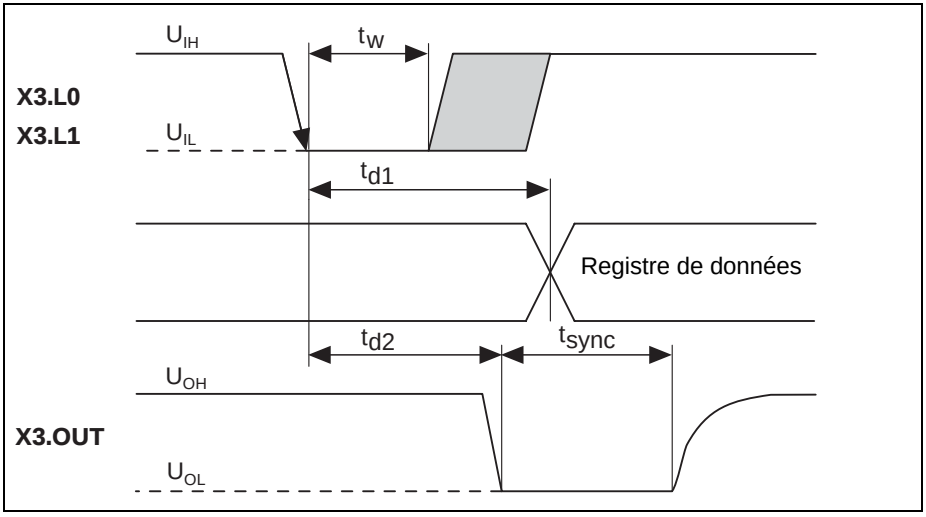
Les entrées X3.L0 et X3.L1 sont actives low; une résistance pull up interne (10 kΩ) les maintient au niveau haut. Elles peuvent être raccordées sur des composants TTL, LS ou CMOS.

Moyen le plus simple pour activer ces entrées: pont allant du 0 volt (plot 4) à l'entrée d'appel.

Sortie d'appel X3.Out
Le signal de sortie X3.Out peut être transmis via l'embase X3 à d'autres IK 121 (entrées X3.L0, X3.L1), par exemple, pour appeler les valeurs de mesure de plusieurs IK.

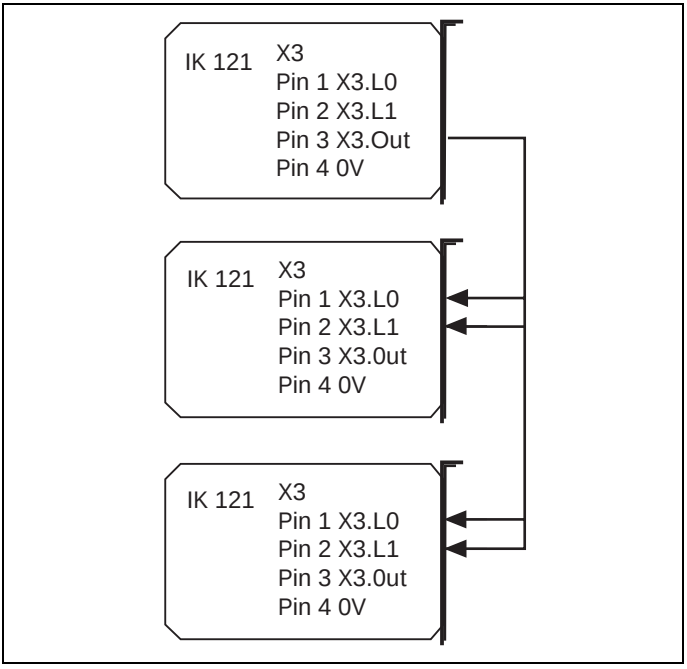
X3.Out est une sortie collecteur ouvert commutant à „zéro".

Entrées d'appel X3.L0, X3.L1 et sortie d'appel X3.0ut: diagramme de temps et niveau de tension



Désignation	**MIN**	**MAX**
U_{IL} (V)	–3,0	0,9
U_{IH} (V)	3,15	30
t_{d1} (µs)	–	24
t_{d2} (ns)	–	500
t_w (ns)	250	–
t_{sync} (µs)	1	–
U_{OL} (V)	0	0,8 (U_{OH}=4V – 12V) 1,0 (U_{OH}=12V – 32V)
U_{OH} (V)	4	32
I_{OL} (mA)	–	40

Appeler les valeurs de mesure de plusieurs IK 121
Par la sortie d'appel X3.Out, il est possible d'appeler les valeurs de mesure à partir de plusieurs IK□21. Pour cela, il convient de r elier la sortie d'appel X3.Out d'une IK□121 avec les autres entré es d'appel des autres IK□121.

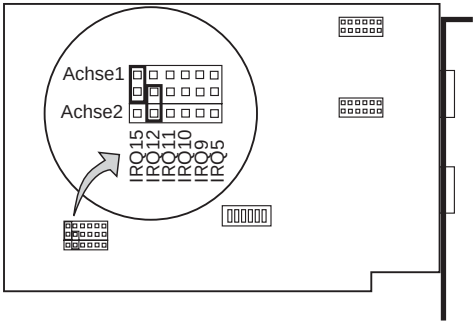


Interruptions

L'IK□121 peut utiliser l'une des interruptions du PC suivantes: I RQ5, IRQ9, IRQ10, IRQ11, IRQ12 ou IRQ15.

L'interruption souhaitée est définie au moyen de ponts réalisés sur la platine.

L'axe 1 génère le signal interne **Int0** et l'axe 2 génère **Int1**. La d isposition des plots permet d'éviter que **Int0** et **Int1** soient con nectés sur la même ligne IRQ.



Sur la configuration ci-dessus, l'axe 1 est sur IRQ15 et l'axe 2, s ur IRQ12.

Affectation des interruptions du PC

Interruption	n° d'interruption	Adresse d'interrup tion
IRQ5	0D	034 à 037
IRQ9	71	1C4 à 1C7
IRQ10	72	1C8 à 1CB
IRQ11	73	1CC à 1CF
IRQ12	74	1D0 à 1D3
IRQ15	77	1D7 à 1DF

Adressage

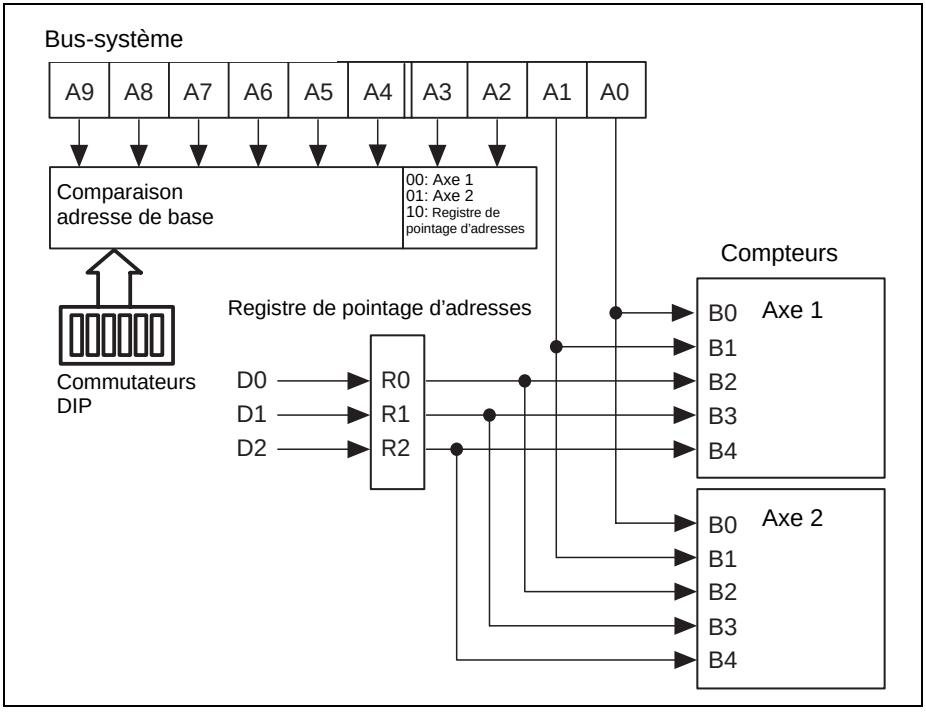
L'IK 121 et le calculateur communiquent au moyen d'adresses de port (plage I/O) de largeur de données 16 bits. L'espace d'ad resse étant limité à l'intérieur de la zone d'adresses de port, les adresses des compteurs ont été superposées. Pour les distingu er, on utilise un registre de pointage d'adresses.

Chaque compteur possède cinq lignes d'adresses B0 à B4.
B0 et B1 sont directement reliées au bus-système. B2, B3 et B 4 sont générés dans le registre de pointage d'adresses.
A2 et A3 du bus-système décodent les compteurs pour l'axe 1, l'axe 2 ou pour le registre de pointage d'adresses.

A4 à A9 du bus-système (adresse de base) sont configurés ave c le commutateur DIP situé sur la platine. Le réglage sur „on" c orrespond à un „zéro" logique.

Exemples de configuration des commutateurs

Adresse	**Configuration des commutateurs**					
(Hex)	A9	A8	A7	A6	A5	A4
0110	on	off	on	on	on	off
0250	off	on	on	off	on	off
0300	off	off	on	on	on	on
0330	off	off	on	on	off	off



Registres

Le synoptique modulaire situé sur la dernière rabat de couvertur e vous aidera dans la compréhension de la description suivante.

Remarque importante concernant la programmation:
L'accès à l'IK 121 a lieu par lecture ou écriture de mots de donn ées dans des registres. C'est pourquoi seules des adresses de port paires peuvent être adressées avec des instructions d'écrit ure de mots et de lecture de mots.

Sommaire des registres

Adresse (Hex) B0 à B4	Accès à l'écriture	Accès à la lecture
00		Registre données 0, mot LSB
02	sans fonction	Registre données 0
04		Registre données 0, mot MSB
06		Registre données 1, mot LSB
08	sans fonction	Registre données 1
0A		Registre données 1, mot MSB
0C Low Byte	Registre d'initialisation 1	Registre d'initialisation 1
High Byte	Registre d'initialisation 2	Registre d'initialisation 2
0E Low Byte	Registre de contrôle 1	Registre d'état 1
High Byte	sans fonction	Registre d'état 2
10 Low Byte	Registre marques de référence	sans fonction
High Byte	sans fonction	Registre valeur d'amplitude
12 Low Byte	Registre de validation pour l'appel de la valeur de mesure	sans fonction
High Byte	Axes en cascade	sans fonction
14 Low Byte	Registre de validation des interru	Registre valid. interruption 1
High Byte	ptions sans fonction	Registre valid. interruption 2
16 Low Byte	Registre d'offset pour signal 0°	Amplitude pour signal 0°
High Byte	sans fonction	Amplitude pour signal 0°
18 Low Byte	Registre d'offset pour signal 90°	Amplitude pour signal 90°
High Byte	sans fonction	Amplitude pour signal 90°
1A Low Byte	Registre de timers, mot LSB	sans fonction
High Byte	Registre de timers, mot MSB	sans fonction
1C Low Byte	Registre de contrôle 2	Registre de contrôle 3
High Byte	sans fonction	Registre d'identification
1E Low Byte	Registre de contrôle 3	Registre d'état 4
High Byte	sans fonction	sans fonction

Registres de données pour les compteurs

Les valeurs de mesure sont mémorisées dans des registres 48 bits. Chaque axe dispose de deux registres de données: regist re de données 0 (00h à 04h) et registre de données 1 (06h à 0Ah). Les valeurs de mesure sont formées à partir de la valeur d'interpolation 10 bits et de la valeur 32 bits du compteur de périodes. Des 48 bits des registres, seuls 42 bits seront utili sés pour la valeur de mesure. Les 6 bits supérieurs sont élargis en représentation du complément à deux en tenant compte du signe.

La largeur des données de 48 bits peut être réduite à 32 bits à l 'aide du registre d'initialisation 1 (0Ch), bit D6.

Au moyen du registre d'initialisation 1 (0Ch), bit D7, on peut en outre définir si la valeur de mesure ne doit être formée qu'à par tir de la valeur du compteur de périodes (bits de données D0 à D9 ne sont pas définis) ou à partir de la valeur du compteu r de périodes et à partir de la valeur d'interpolation.
La mémorisation des valeurs de comptage dans les registres de données peut s'effectuer par:

- appel de logiciel
- entrées externes
- timers
- marques de référence

L'action des différents signaux d'appel est illustrée dans le syno ptique modulaire des compteurs situé sur le dernier rabat de la brochure.

Vous pouvez interroger le registre d'état 1 (0Eh), bit D0 ou D1 p our savoir si la valeur de mesure a été mémorisée dans les regi stres de données. Tant que le bit D0 ou D1 est initialisé, aucune autre valeur de mesure ne peut être mémorisée jusqu'à ce que le mot supérieur de la valeur de mesure ait été lu (exception: p ar le registre de contrôle 2, bit D6 ou D7, l'appel est validé sans que la valeur de mesure ait été précédemment lue) En mode 48 bits, il s'agit des registres de données 04h ou 0Ah, et en mode 32 bits, des registres de données 02h ou 08h. Une fois la lectu re de la valeur de mesure achevée, le bit D0 ou D1 du registre d 'état 1 (0Eh) est désactivé.

Si le compteur est arrêté ou s'il est mémorisé par franchisse ment de la marque de référence, les bits D0 à D9 comportent alors la valeur fixe 256.

0Ch: Registre d'initialisation 1 (accès à l'écriture)

Bit	Fonction
D0	**Fonctionnement avec/sans interpolation** 0 = Fonctionnement comme compteur de périodes (sans interpolation – bits de données D0 à D9 ne sont pas définis) 1 = Valeur de mesure formée à partir de la valeur du compteur de périodes et à partir de la valeur d'interpolation.
D1	sans fonction
D2	**Timers** 0 = Annuler et arrêter le timer 1 = Lancer le timer
D3	sans fonction
D4	
D5	
D6	**Validation de la mémorisation** 0 = Mode: registre 32 bits Lecture des bits D24 à D31 annule le bit d'état D0 ou D1 dans le registre d'état 1 (0Eh). 1 = Mode: registre 48 bits Lecture des bits D40 à D47 annule le bit d'état D0 ou D1 dans le registre d'état 1 (0Eh).
D7	**Sens de comptage** Le sens de comptage définit si les compteurs comptent dans le sens de déplacement positif (normal) ou négatif (inversé). 0 = Sens de comptage normal 1 = Sens de comptage inversé Le sens de comptage inversé n'est autorisé qu'en mode compteur de périodes! En fonctionnement avec interpolation, le sens de comptage inverse une liaison erronée de la valeur d'interpolation et de la valeur du compteur de périodes.

0Ch: Registre d'initialisation 2 (accès à l'écriture)

Bit	Fonction
D8 D9	**Uniquement en mode compteur de périodes: exploitation des fronts** Les deux signaux incrémentaux du système de mesure (0° él. et 90° él.) fournissent pour chaque période du signal quatre fronts max. permettant leur exploitation. Les compteurs peuvent être programmés de manière à ce qu'ils comptent un, deux ou quatre fronts par période de signal. D9 D8 0 0 = par 1 1 0 = par 2 1 1 = par 4 En mode avec valeur d'interpolation, l'exploitation par 1 est automatiquement configurée.
D10	**Uniquement en mode compteur de périodes: mode de comptage** 0 = Comptage linéaire -2^{41} à $+2^{41}-1$ 1 = Comptage angulaire tel que défini dans D11 Pour systèmes de mesure angulaire avec 36 000 ou 360 000 traits par tour.
D11	**Uniquement avec affichage angulaire: mode de comptage** 0 = 17 999 à –18 000 1 = 179 999 à –180 000
D12 **D13**	sans fonction
D14 **D15**	**Appel de la valeur de mesure avec impulsion de référence** 0 = 1ère marque de référence mémorisée dans le registre de données 0 1 = 1ère marque de référence mémorisée dans le registre de données 1 0 = 2ème marque de référence mémorisée dans le registre de données 0 1 = 2ème marque de référence mémorisée dans le registre de données 1

0Ch: Accès à l'écriture: bits D0 à D15: retour lecture des registres d'initialisation 1 et 2

0Eh: Registre de contrôle 1 (accès à l'écriture)

Bit	Fonction
D0	1 = Appel de logiciel: valeur de mesure dans le registre de données 0
D1	1 = Appel de logiciel: valeur de mesure dans le registre de données 1
D2	1 = Appel de logiciel dans tous les registres (doit être validé dans le registre de validation d'appel)
D3	1 = Lancer le compteur
D4	1 = Stopper le compteur
D5	1 = Effacer le compteur
D6	1 = Effacer défaut du système de mesure (dépassement de fréquence)
D7	Effacer le registre de valeur d'amplitude
D8 D9 D10 D11 D12 D13 D14 D15	sans fonction

0Eh: Registre d'état 1 (accès à la lecture)

Bit	Fonction
D0	Etat pour appel de logiciel dans le registre 0 1 = Valeur de mesure prête
D1	Etat pour appel de logiciel dans le registre 1 1 = Valeur de mesure prête
D2	sans fonction
D3	sans fonction
D4	1 = Compteur stoppé
D5	Compteur 1: sans fonction Compteur 2: ligne SDA bus I²C (entrée)
D6	1 = Défaut système de mesure (dépassement de fréquence)
D7	sans fonction

0Eh: Registre d'état 2 (accès à la lecture)

Bit	Fonction
D8	1 = Franchissement des marques de référence actif
D9 D10 D11 D12	sans fonction
D13	Niveau logique pour le signal 0°
D14	Niveau logique pour le signal 90°
D15	Niveau logique de la marque de référence

10h: Registre des marques de référence (accès à l'écriture)

Les systèmes de mesure linéaire et angulaire HEIDENHAIN peuvent avoir une ou plusieurs marques de référence. HEIDENHAIN préconise tout particulièrement les systèmes de mesure avec marques de référence à distances codées.

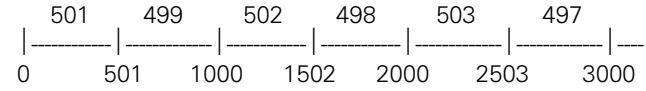
Lors d'une coupure de courant, la relation entre la position du système de mesure et la valeur d'affichage est perdue. Grâce aux marques de référence des systèmes de mesure, cette relation est rétablie lors de la remise sous tension.

Lors du franchissement des marques de référence, un signal est délivré. Ce signal désigne cette position de la règle de mesure comme point de référence. Celui-ci permet de rétablir les relations entre les positions sur les axes et les valeurs d'affichage précédemment définies. Sur les systèmes de mesure linéaire avec marques de référence à distances codées, il suffit pour cela d'effectuer un déplacement de 20 mm.

Bit	Fonction au franchissement de la marque de référence
D0	1 = Lancer le compteur
D1	1 = Stopper le compteur
D2	1 = Effacer le compteur
D3	1 = Appeler la valeur de mesure
D4	1 = Appeler la valeur de mesure au franchissement de la 2ème marque de référence
D5	1 = Effacer le compteur à chaque franchissement d'une marque de référence
D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	sans fonction

Exploitation des marques de référence à distances codées
Sur les systèmes de mesure équipés de marques de référence à distances codées, des marques de référence sont situées à é carts fixes sur toute la course de mesure. Entre deux marques de référence, on trouve une troisième marque dont l'écart par r apport aux deux autres est variable de telle sorte que chaque éc art soit un multiple de la période de division et qu'il ne puisse s e produire qu'une seule fois sur toute la course de mesure. Apr ès une coupure de courant, il suffit donc de franchir deux marq ues de référence pour rétablir les relations entre les positions s ur les axes et les valeurs d'affichage.

L'exemple suivant illustre l'exploitation de ces marques de référ ence à distances codées:
Sur une distance de base de 1 000 périodes de signal, on a la ré partition suivante en incréments pour les marques de référence :



Le registre de marques de référence 10h doit tout d'abord être i nitialisé de la manière suivante:

- Effacer le compteur avec le franchissement de la 1ère marq ue de référence et le lancer (bit D0 = 1 et D2 =1).
- Appeler le compteur avec le franchissement de la 2ème mar que de référence (bit D4 = 1).

Pour calculer la position absolue, il suffit de disposer de l'écart en périodes de signal entre deux marques de référence. Cet éc art en incréments – désigné dans la description suivante par DI FF – doit être divisé par 1□024.

DIFF = DISTANCE_IN_INCR : 1 024

Pour calculer la position absolue, il faut calculer le décalage (OF FSET) entre la 1ère marque de référence sur la règle de mesure (position „0“ sur le plan) et la 1ère marque de référence franch ie.

Quatre cas sont possibles:

1. Sens de déplacement positif et DIFF > 500
   OFFSET = (DIFF – 501) • 1 000
2. Sens de déplacement positif et DIFF < 500
   OFFSET = (500 – DIFF) • 1 000 – DIFF
3. Sens de déplacement négatif et |DIFF| > 500
   OFFSET = (DIFF – 501) • 1 000 + DIFF
4. Sens de déplacement négatif et |DIFF| < 500
   OFFSET = (500 – DIFF) • 1 000

La position absolue en incréments est calculée de la manière su ivante:

ABS_POS_INCR = ACT_POS + PRESET + DISTANCE

ACT_POS: écart entre la position actuelle et la 1ère marque de r éférence franchie (en incréments).

PRESET: position initialisée (en incréments) lors de l'initialisatio n du point de référence sur la 1ère marque de référence de la r ègle de mesure (position „0“ sur le plan).

DISTANCE: écart entre la 1ère marque de référence sur la règle de mesure et la 1ère marque de référence franchie (en incrém ents).

DISTANCE = OFFSET • 1 024

La position absolue calculée – par ex. sur une règle de mesure avec une période de signal de 0,02 mm – de la manière suivant e:

$$\text{ABS\_POS\_MM} = \frac{\text{ABS\_POS\_INCR} \bullet 0{,}02}{1\,024}$$

Sur systèmes de mesure angulaire:

$$\text{ABS\_POS\_DEGREE} = \frac{\text{ABS-POS\_INCR} \bullet 360°}{1\,024 \bullet \text{LINES\_PER\_REV.}}$$

Vous trouverez un exemple d'application en „TURBO PASCAL“ de l'exploitation des marques de référence dans le code sourc e de TNC.EXE et dans le fichier CNT_2.PAS sous (* Distance-co ded ref. marks *).

10h: Registre de valeur d'amplitude (accès à la lecture)

Bit	Fonction
D0 D1 D2 D3 D4 D5 D6 D7	sans fonction
D8 D9	**Amplitude actuelle** Une nouvelle valeur d'amplitude est communiquée à c haque appel d'une valeur de mesure. D9 D8 **IK 121 A** **IK 121 V** 0 0 amplitude normale 5 µA < I_e < 15 µA 0,47 V_{CC} < U_e < 1,41 V_{CC} 0 1 amplitude faible 2,5 µA < I_e < 5 µA 0,23 V_{CC} < U_e < 0,47 V_{CC} 1 0 grande amplitude I_e > 15 µA U_e > 1,41 V_{CC} 1 1 petite amplitude erronée I_e < 2,5 µA U_e < 0,23 V_{CC} Avant la lecture, la valeur d'amplitude devrait être „gel ée“ par le bit D4 dans le registre de contrôle 2. Le registre de valeur d'amplitude est annulé par le bit D7 dans le registre de contrôle 1.
D10 D11	**Valeur min. de l'amplitude** Codage et accès à la lecture: cf. bits D8 et D9 Si la valeur d'amplitude est inférieure à la valeur min. mémorisée et ce, plus de quatre fois successivement, l'IK remplace l'ancienne valeur min. par la valeur d'amp litude actuelle.
D12 D13 D14 D15	sans fonction

12h: Registre de validation pour l'appel de la valeur de mesure (accès à l'écriture)

Bit	Fonction
D0	Validation de L0 pour registre de données 0 Axe 1: 0 Axe 2: 1 = Validation des axes en cascade avec l'axe 1
D1	Validation de L0 par circuit de retard (125 ns) pour registre de données 0 Axe 1: 1 = Validation du signal d'appel externe m X3.L0 pour registre de données 0 Axe 2: 0
D2	1 = Validation de l'„appel de logiciel dans tous les registres de données" pour registre de données 0
D3	1 = Validation de l'„appel de logiciel par timer" pour registre de données 0
D4	Validation de L1 pour registre de données 1 Axe 1: 1 = Validation du signal d'appel externe X3.L0 pour registre de données 1 Axe 2: 1 = Validation du signal d'appel externe X3.L1 pour registre de données 1
D5	Validation de L1 par circuit de retard (125 ns) pour registre de données 1 0
D6	1 = Validation de l'„appel de logiciel dans tous les registres de données" pour registre de données 1
D7	1 = Validation de l'„appel de logiciel par timer" pour registre de données 1

Le synoptique modulaire situé sur dernier rabat de cette brochure illustre la fonction des différents bits.

12h: Registre des axes en cascade et de commande du bus I²C (accès à l'écriture)

Bit	Fonction	
D8	Axe 1: Axe 2:	1 = Validation du signal d'appel externe pour le 2ème axe (L0, X3.Out) 0
D9	Axe1: Axe 2:	1 = Validation de l'„appel de logiciel dans tous les registres de données" pour le 2ème axe (L0, X3.Out) 0
D10	Axe 1: Axe 2:	1 = Validation du strobe de timer pour le 2ème axe (L0, X3.Out) 0
D11	Axe 1: Axe 2:	0 Ligne SDA du bus I²C: données (signaux à programmer de manière inverse)
D12	sans fonction	
D13		
D14		
D15	Axe 1: Axe 2:	0 Ligne SCL du bus I²C: horloge (signaux à programmer de manière inverse)

Le synoptique modulaire situé sur dernier rabat de cette brochu re illustre la fonction des différents bits.

12h: Accès à la lecture sans fonction

14h: Registre de validation des interruptions (accès à l'écriture)

La logique d'interruption est programmée au moyen du registre de validation d'interruption. L'interruption peut être causée par l'appel de la valeur de mesure dans le registre 0, le registre 1 ou le strobe de timer. Chacune de ces trois sources d'interruption peuvent programmées indépendamment des autres. Si plusieurs interruptions interviennent simultanément, la priorité est la suivante:

- Priorité max.: appel valeur de mesure par registre 0
- Deuxième priorité: appel valeur de mesure par registre 1
- Priorité min.: appel valeur de mesure par strobe de timer

Après une interruption, aucune autre ne peut intervenir tant que le registre d'état interruption 2 n'a pas été lu.

Bit	Fonction
D0	1 = Validation de l'interruption 0 lors de l'appel de la valeur de mesure avec registre 0
D1	1 = Validation de l'interruption 1 lors de l'appel de la valeur de mesure avec registre 1
D2	1 = Validation de l'interruption 2 pour strobe de timer
D3	1 = Interruption sera générée (D0 = D1 = D2 = 0)
D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	sans fonction

Le synoptique modulaire situé sur dernier rabat de cette brochure illustre la fonction des différents bits.

14h: Registre d'état interruption 1 (accès à la lecture)

L'interruption actuelle qui est active est affichée dans ce registr e (on ne peut activer qu'un seul bit à la fois). La lecture de ce re gistre provoque l'annulation de l'interruption actuelle qui est acti ve et efface le bit d'état correspondant de manière à ce que l'in terruption suivante puisse être déclenchée par un front négatif.

Bit	Fonction
D0	1 = Interruption 0 active, un appel de la valeur de mesure a eu lieu avec le registre de données 0
D1	1 = Interruption 1 active, un appel de la valeur de mesure a eu lieu avec le registre de données 1
D2	1 = Interruption 2 active, un appel de la valeur de mesure a eu lieu avec le strobe de timer
D3 D4 D5 D6 D7	sans fonction

14h: Registre d'état interruption 2 (accès à la lecture)

Tous les interruptions en instance sont affichées dans ce registr e, c'est-à-dire l'interruption active et les interruptions qui resten t à exécuter. Plusieurs bits peuvent donc être activés simultané ment.

Bit	Fonction
D8	1 = Interruption 0 en instance mais non encore exécutée
D9	1 = Interruption 1 en instance mais non encore exécutée
D10	1 = Interruption 2 en instance mais non encore exécutée
D11 D12 D13 D14 D15	sans fonction

16h: Registre d'offset pour signal 0° (accès à l'écriture)
Ce registre contient la valeur de correction d'offset 7 bits pour l e signal 0° en représentation à complément à deux. Il en ressor t une correction max. de ± 63.
Les valeurs de correction ne peuvent être écrites que si l'un de s bits d'état D5 ou D6 a la valeur 0 dans le registre d'état 3.
Principe fonctionnel:
Des valeurs de correction d'offset sont ajoutées aux valeurs dig itales du signal 0° (0 à 1023) et du signal 90°. Lors d'un déplace ment, la valeur est limitée à 1023 ou à 0.

Bit	Fonction
D0 D1 D2 D3 D4 D5 D6	Valeur de correction d'offset pour le signal 0° en représentation à complément à deux
D7 D8 D9 D10 D11 D12 D13 D14 D15	sans fonction

Le registre d'offset dans les compteurs **n'est pas** protégé en cas de coupure d'alimentation. Pour cette raison les valeurs d e correction d'offset sont mémorisées dans une EEPROM du composant pour les potentiomètres électroniques. A la mise s ous tension, les valeurs de correction d'offset doivent être ch argées de l'EEPROM dans les registres d'offset des compteu rs (cf. procédures „store_offset" et „load_offset").

16h: Amplitude pour le signal 0° (accès à la lecture)
A chaque conversion analogue-digitale, le résultat est mémorisé . Avant la lecture, les valeurs doivent être "gelées" avec le bit D 4 dans le registre de contrôle 2.

Bit	Fonction
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9	Amplitude pour le signal 0°
D10 D11 D12 D13 D14 D15	sans fonction

18h: Registre d'offset pour le signal 90° (accès à l'écriture)
Ce registre contient la valeur de correction d'offset 7 bits pour l e signal 90°.
Description du principe fonctionnel: cf. „16h: Registre d'offset pour le signal 0°"

Bit	Fonction
D0 D1 D2 D3 D4 D5 D6	Valeur de correction d'offset pour le signal 90° en représentation du complément à deux
D7 D8 D9 D10 D11 D12 D13 D14 D15	sans fonction

18h: Amplitude pour le signal 90° (accès à la lecture)
A chaque conversion analogue-digitale, le résultat est mémorisé . Avant la lecture, les valeurs doivent être "gelées" avec le bit D 4 dans le registre de contrôle 2.

Bit	Fonction
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9	Amplitude pour le signal 90°
D10 D11 D12 D13 D14 D15	sans fonction

1Ah: Registre de timer (accès à l'écriture)
La valeur de timer 13 bits est incrite dans les registres 1Ah et 1Bh. Des valeurs de 0 à 8191 peuvent ainsi être mémorisées dans le reg istre de timer. La durée de cycle est donnée en ns; il convient toute fois de soustraire 1 µs à la durée de cycle désirée.

Exemple:
Durée de cycle désirée = 1 ms = 1 000 µs
Valeur à programmer = 1 000 – 1 = 999
Pour lancer le timer, il ne suffit pas de l'écrire. Le lancement du timer s'effectue en initialisant le bit D2 dans le registre d'initialis ation 1 (0Ch). Il faut également initialiser les bits correspondant s dans les registres de validation 12h, 13h ou 14h.

Bit	Fonction
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12	Valeur du timer
D13 D14 D15	sans fonction

1Ah: Accès à la lecture sans fonction

1Ch: Registre de contrôle 2 (accès à l'écriture)

Bit	Fonction
D0 D1 D2 D3	Programmer une valeur fixe ≥ 8
D4	1 = "Geler" l'amplitude pour le signal 0° (16h) et le signal 90° (18h) ainsi que le registre de valeur d'amplitude (10h High Byte). Afin d'éviter toute modification des valeurs de registres pendant la lecture, il convient d'initialiser ce bit. Pour savoir si les valeurs de registres sont "gelées", il convient d'interroger le bit D4 dans le registre d'état 3.
D5	0
D6	1 = Nouvel appel possible avec registre de données 0 sans devoir chercher préalablement la valeur de mesure. Avec ce mode, il est possible de modifier la valeur de mesure pendant la lecture.
D7	1 = Nouvel appel possible avec registre de données 1 sans devoir chercher préalablement la valeur de mesure. Avec ce mode, il est possible de modifier la valeur de mesure pendant la lecture.
D8 D9 D10 D11 D12 D13 D14 D15	sans fonction

1Ch: Registre d'état 3 (accès à la lecture)

Bit	Fonction
D0 D1 D2 D3	illisible
D4	1 = L'amplitude pour le signal 0° (16h) et le signal 90° (18h) ainsi que le registre de valeur d'amplitude (10h High Byte) sont "gelés" et peuvent être lus.
D5	0 = Le registre d'offset pour le signal 0° a été écrit.
D6	0 = Le registre d'offset pour le signal 90° a été écrit.
D7	sans fonction

1Ch: Registre d'identification (accès à la lecture)

Bit	Fonction
D8 D9 D10 D11 D12 D13 D14 D15	Identification du circuit: valeur fixe 8

1Eh: Registre de contrôle 3 (accès à l'écriture)

Bit	Fonction
D0 D1	Valeur fixe 0
D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	sans fonction

1Eh: Registre d'état 4 (accès à la lecture)

Bit	Fonction
D0	sans fonction
D1	Niveau logique sur plot L0
D2	Niveau logique sur plot L1
D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	sans fonction

L'IK 121 dans les applications DOS

Accès rapide au premier affichage

Ne pas monter tout de suite l'IK 121.

- ➤ Insérez la disquette avec la logiciel driver dans le lecteur A de votre PC.
- ➤ Introduisez les instructions DOS suivantes:
  >A:
  >INSTALL
  Après l'écran d'accueil, votre PC affiche la remarque suivante:

```
INSTALL IK 121
The IK 121-Interface-card should not be in
your PC!
Continue (y/n)?
```

- ➤ Introduisez „y".
  Le programme INSTALL affiche maintenant le sommaire des adresses de port.
  Les adresses de port libres sont désignées avec „free" et les adresses de port occupées avec „not free".
- ➤ Sélectionnez une adresse de port libre et introduisez le numéro correspondant: par exemple „4".
  INSTALL vous indique maintenant comment vous devez configurer les commutateurs DIP sur l'IK 121.

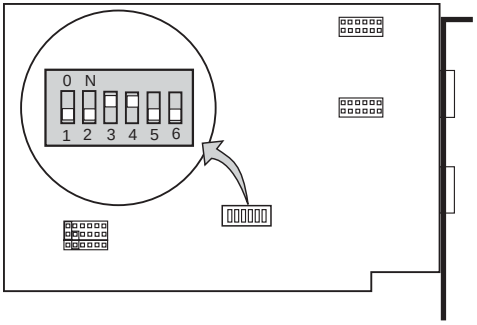
```
Chosen address:

   Nr:4 Adr.:0330 free


The DIP switch position on your interface card is:

   A4--off
   A5--off
   A6--on
   A7--on
   A8--off
   A9--off
                          on
                          A4          A9
                                        Press Return
```

➤ Configurez les commutateur DIP sur l'IK 121.



➤ Sur votre PC, appuyez sur „Return". Le PC mémorise dans le fichier IK 121.INI l'adresse de port sélectionnée. L'adresse est disponible pour tous les exemples de programmation fournis – excepté pour les exemples simples SAMPLE32.PAS, SAMPLE48.PAS, SAMPLE32.C et SAMPLE48.C –.

➤ Appuyez sur „Return".
Ceci met un terme au programme INSTALL

Si le programme INSTALL ne fonctionne pas sur votre PC, pr océdez de la manière suivante:

- Utilisez la configuration standard (adresse 0330h). Celle-ci correspond à la fois à la configuration en usine de la platine IK□121 et à la pré-configuration pour le logiciel. Si cette adresse est déjà occupée sur votre PC,
- à l'aide d'un éditeur de texte, inscrivez dans le fichier IK121.INI l'adresse que vous avec choisie (en décimal!) ou
- lancez le programme IK121.EXE et sélectionnez „File", „Setup IK121", „Setup board", „Base address" et introduisez l'adresse désirée (en décimal!). Mémorisez l'adresse à l'aide de „Save setup".

➤ Retirez la disquette du lecteur A, débranchez la prise secteur de votre PC et ouvrez le carter. Si vous ignorez comment insérer de nouvelles cartes sur votre ordinateur, reportez-vous au mode d'emploi de votre PC.

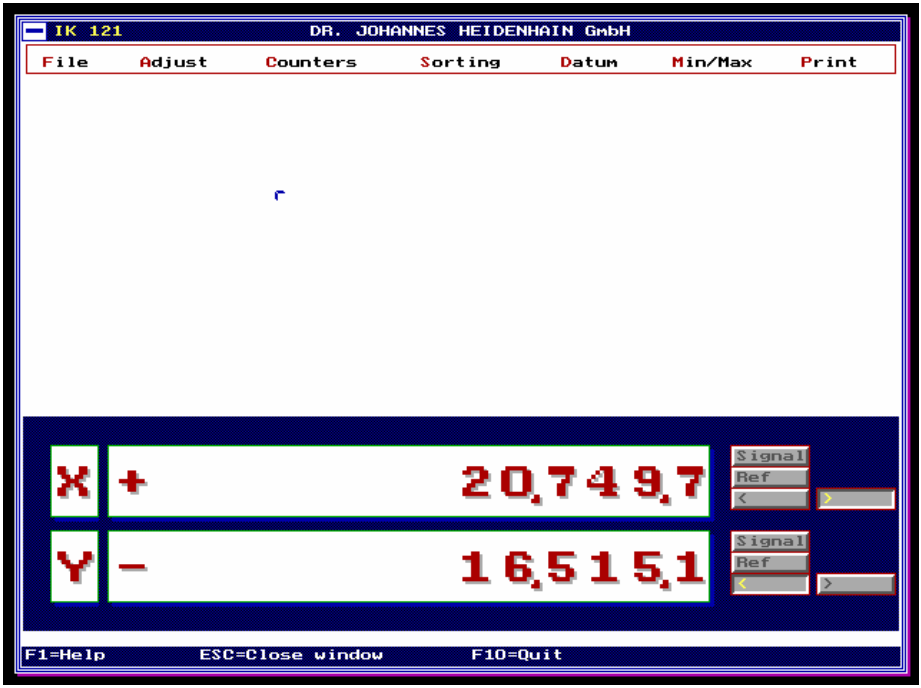
➤ Insérez l'IK 121 dans votre PC (un slot court suffit), fixer la carte et remonter le carter de votre PC.

➤ Raccorder un ou deux système(s) de mesure délivrant des signaux de sortie sinusoïdaux sur les entrées „X1" ou „X2" en utilisant le câble adaptateur HEIDENHAIN (Id.-Nr. 309 785 ..).

➤ Enficher le câble secteur sur le PC et mettre celui-ci sous tension.

➤ Insérer à nouveau la disquette avec le logiciel driver dans le lecteur A de votre PC.
Introduisez les instructions DOS suivantes:
>A:
>IK121

➤ Après l'écran d'accueil, votre PC affiche:



Le programme IK 121.EXE affiche les valeurs effectives de positions des systèmes de mesure raccordés.

Pour les exemples de programmation dont l'utilisation est conversationnelle, vous pouvez vous servir soit de la souris, soit du clavier. A l'aide de la touche F1, vous avez accès à plus amples informations concernant les différents menus.

Ce Manuel décrit la programmation de l'IK 121 dans le applications DOS illustrée par des exemples en „TURBO PASCAL" et en „BORLAND C".
L'IK 121 utilise des adresses de port et peut être programmée dans n'importe quel langage de programmation capable de traiter des adresses de ports.

Les fichiers IK121_0.*: Fonctions de base pour l'écriture et la lecture des registres

Vous découvrirez les fonctions décrites ci-après sur la disquette 1 contenue dans la fourniture. Pour „TURBO PASCAL", sous le répertoire **TP** dans le fichier **IK121_0.PAS**. Pour „BORLAND C" ; sous le répertoire **BC** dans le fichier **IK121_0.C** et dans le fichier header correspondant **IK121_0.H**.
Les compteurs de l'IK 121 sont adressés par des registres d'adresses (cf. „Adressage"). Le registre d'adresses doit avoir été initialisé avant qu'un compteur puisse être adressé. Les deux fonctions **write_g26** et **read_g26** permettent d'écrire et de lire les registres de données; ce sont les fonctions de base pour travailler avec l'IK 121.

Différence entre „TURBO PASCAL" et „BORLAND C"

„TURBO PASCAL" reconnaît les **procédures** (fonctions sans valeur de renvoi) et **fonctions** (avec valeur de renvoi) alors que „BORLAND C" ne reconnaît que les fonctions (avec ou sans valeur de renvoi). Cette description utilise les désignation du „TURBO PASCAL" „**procédure**" et „**fonction**".

Procédure d'écriture dans les registres

La procédure suivante permet d'écrire une valeur dans un registre 16 bits d'un compteur.

Procédure: **write_g26**[1)]

Paramètres:

baseadr:	Adresse de base de l'IK 121 configurée avec les commutateurs DIP
axis:	1 = axe 1, 2 = axe 2
address:	Adresses B0 à B4 du registre du compteur
datum:	Valeur devant être écrite sur l'adresse

1) g26 correspond à la désignation HEIDENHAIN du compteur

Procédure en „TURBO PASCAL"

```
PROCEDURE
 write_g26(baseadr:word;axes,address:byte;datum:word);
  VAR adr_reg,adr_point,adr_gate : word;

  BEGIN
       (*Supprimer les quatre derniers bits de l'adresse de
       carte*)
    baseadr:=baseadr and $0FF0;
       (*Adresse B0 à B4 du compteur*)
    address:=address and $001F;

    (*Charger le pointeur d'adresse dans le registre
    d'adresses*)
       (*Adresse du registre d'adresses*)
    adr_reg:=baseadr or $0008;
       (*Contenu du registre d'adresses R0 à R2 =
         adresse du compteur sans B0 et B1*)
    adr_point:=address shr 2;
       (*Charger le registre d'adresses*)
    portw[adr_reg]:=adr_point;

    (*Calculer l'adresse de port*)
    if axes=1 then
      adr_gate:=baseadr or (address and $03)
    else
      adr_gate:=(baseadr or $0004) or (address and $03);

       (*Ecrire les données dans le compteur*)
    portw[adr_gate]:=datum;
  END;
```

Procédure en „BORLAND C"

```
void write_g26 (unsigned int baseadr, unsigned char axis,
                unsigned int address, unsigned int datum)
{
unsigned int          adr_reg, adr_point, adr_gate;
   /*Supprimer les quatre derniers bits de l'adresse de
     carte*/
baseadr = baseadr & 0x0FF0;
   /*Adresse B0 à B4 du compteur*/
address = address & 0x1F;

/*Charger le pointeur d'adresse dans le registre
  d'adresses*/
   /*Adresse du registre d'adresses*/
adr_reg = baseadr | 0x0008;
   /*Contenu du registre d'adresses R0 à R2 =
     adresse du compteur sans B0 et B1*/
adr_point = address >> 2;
   /*Charger le registre d'adresses*/
outpw (adr_reg, adr_point);

/*Calculer l'adresse de port*/
switch (axis)
       {
       case 1:
       adr_gate = baseadr | (address & 0x03);
       break;
       case 2:
       adr_gate = (baseadr | 0x0004) | (address & 0x03);
       break;
       }

   /*Ecrire les données dans le compteur*/
outpw (adr_gate, datum);
}
```

Fonction de lecture des registres
La fonction suivante permet de lire une valeur à partir d'un regis tre 16 bits d'un compteur.

Fonction: **read_g26**[1)]

Paramètres

baseadr:	Adresse de base de l'IK 121 configurée avec les commutateurs DIP
axis:	1 = axe 1, 2 = axe 2
address:	Adresse B0 à B4 du registre du compteur
résultat:	Valeur du registre de données indiquée sous „axis" et „address" comme variable 16 bits

Fonction en „TURBO PASCAL"

```
FUNCTION read_g26(baseadr:word;axes,address:byte):word;
  VAR adr_reg,adr_point,adr_gate : word;

  BEGIN
       (*Supprimer les quatre derniers bits de l'adresse de
         carte*)
     baseadr:=baseadr and $0FF0;
        (*Adresse B0 à B4 du compteur*)
     address:=address and $001F;

     (*Charger le pointeur d'adresse dans le registre
      d'adresses*)
        (*Adresse du registre d'adresses*)
     adr_reg:=baseadr or $0008;
        (*Contenu du registre d'adresses R0 à R2 =
         adresse du compteur sans B0 et B1*)
     adr_point:=address shr 2;
        (*Charger le registre d'adresses*)
     portw[adr_reg]:=adr_point;

     (*Calculer l'adresse de port*)
     if axes=1 then
       adr_gate:=baseadr or (address and $03)
     else
       adr_gate:=(baseadr or $0004) or (address and $03);

        (*Lire les données dans le compteur*)
     read_g26:=portw[adr_gate];
  END;
```

1) g26 correspond à la désignation HEIDENHAIN du compteur

Fonction en „BORLAND C"

```
unsigned int read_g26 (unsigned int baseadr,
                unsigned char axis, unsigned int address)
{
unsigned int adr_reg, adr_point, adr_gate;
   /*Supprimer les quatre derniers bits de l'adresse de
     carte */
baseadr = baseadr & 0x0FF0;
   /*Adresse B0 à B4 du compteur*/
address = address & 0x1F;

/*Charger le pointeur d'adresse dans le registre
  d'adresses */
   /*Adresse du registre d'adresses*/
adr_reg = baseadr | 0x0008;
   /*Contenu du registre d'adresses R0 à R2 =
     adresse du compteur sans B0 et B1*/
adr_point = address >> 2;
   /*Charger le registre d'adresses*/
outpw (adr_reg, adr_point);

/*Calculer l'adresse de port*/
switch (axis)
       {
       case 1:
       adr_gate = baseadr | (address & 0x03);
       break;
       case 2:
       adr_gate = (baseadr | 0x0004) | (address & 0x03);
       break;
       }
   /*Lire les données dans le compteur*/
return(inpw(adr_gate));
}
```

Fonctions de base permettant d'appeler la valeur de mesure par le logiciel

Procédures de mémorisation d'une valeur de mesure
Les procédures suivantes permettent de mémoriser la position du compteur de l'axe désiré dans le registre de données 0 (soft _l0) ou le registre de données 1 (soft_l1).

Procédure:	**soft_l0** **soft_l1**
Paramètres	
baseadr:	Adresse de base de l'IK 121 configurée avec les commutateurs DIP
axis:	1 = axe 1, 2 = axe 2

Procédures en „TURBO PASCAL"

```
PROCEDURE soft_l0(baseadr:word;axis:byte);
  BEGIN
    write_g26(baseadr,axis,14,$0001);
  END;

PROCEDURE soft_l1(baseadr:word;axis:byte);
  BEGIN
    write_g26(baseadr,axis,14,$0002);
  END;
```

Procédure en „BORLAND C"

```
void soft_l0 (unsigned int baseadr, unsigned char axis)
{
write_g26 (baseadr, axis, 0x0e, 0x0001);
}
void soft_l1 (unsigned int baseadr, unsigned char axis)
{
write_g26 (baseadr, axis, 0x0e, 0x0002);
}
```

Fonctions permettant de contrôler si la valeur de mesure a bien été mémorisée
Fonction: latched

Paramètres

baseadr:	Adresse de base de l'IK 121 configurée avec les commutateurs DIP
axis:	1 = axe 1, 2 = axe 2
reg:	0 = registre de données 0 1 = registre de données 1
résultat:	false = aucune valeur de mesure mémorisée true = une valeur de mesure mémorisée

Fonction en „TURBO PASCAL"

```
FUNCTION latched(baseadr:word;axis,reg:byte):boolean;
  BEGIN
      case reg of
        0: latched:=
        (Read_g26(baseadr,axis,14) and $0001 ) = $0001;
        1: latched:=
        (Read_g26(baseadr,axis,14) and $0002 ) = $0002;
      end;
  END;
```

Fonction en „BORLAND C"

```
unsigned char latched (unsigned int baseadr,
                  unsigned char axis, unsigned char reg)
{
unsigned char result;
switch (reg)
       {
       case 0:
       result = (unsigned char)
                (read_g26 (baseadr, axis, 14) & 0x0001);
       break;
       case 1:
       result = (unsigned char)
                (read_g26 (baseadr, axis, 14) & 0x0002);
       break;
       }
return (result);
}
```

Procédure permettant de contrôler à nouveau si la valeur d e mesure a bien été mémorisée

La procédure suivante renouvelle l'interrogation jusqu'à ce qu'u ne valeur de mesure ait été mémorisée.

Procédure: **poll_latch**

Paramètres

baseadr:	Adresse de base de l'IK 121 configurée avec les commutateurs DIP
axis:	1 = axe 1, 2 = axe 2
reg:	0 = registre de données 0 1 = registre de données 1

Fonction en „TURBO PASCAL"

```
PROCEDURE poll_latch(baseadr:word;axis,reg:byte);
  BEGIN
      case reg of
        0: begin
             repeat
             until latched(baseadr,axis,0);
           end;
        1: begin
             repeat
             until latched(baseadr,axis,1);
           end;
      end;
  END;
```

Fonction en „BORLAND C"

```
void poll_latch (unsigned int baseadr,unsigned char axis,
                 unsigned char reg)
{
switch (reg)
       {
       case 0:
       while (latched (baseadr, axis, 0) == 0)
       ;
       break;

       case 1:
       while (latched (baseadr, axis, 1) == 0)
       ;
       break;
       }
}
```

Fonction de lecture d'une valeur de mesure 32-bits
La fonction suivante permet de lire dans un compteur une valeu r de mesure 32□bits.

Fonction: **read_count_value32**

Paramètres

baseadr:	Adresse de base de l'IK 121 configurée avec les commutateurs DIP
axis:	1 = axe 1, 2 = axe 2
reg:	0 = registre de données 0 1 = registre de données 1
résultat:	„TURBO PASCAL": Valeur du registre de données indiquée sous **axis** et **reg** comme variable entière de type **comp** „BORLAND C": comme variable entière de type **long**

Fonction en „TURBO PASCAL"

```
FUNCTION
 read_count_value32(baseadr:word;axis,reg:byte):comp;
  TYPE
    vartype = (li,by);
    mapper = record
               case wert:vartype of
                 li : (field0:longint);
                 by : (field1:array[0..1] of word);
               end;
  VAR
    buffer : mapper;
  BEGIN
    case reg of
     0 : begin
           buffer.field1[0]:=read_g26(baseadr,axis,0);
           buffer.field1[1]:=read_g26(baseadr,axis,2);
         end;
     1 : begin
           buffer.field1[0]:=read_g26(baseadr,axis,6);
           buffer.field1[1]:=read_g26(baseadr,axis,8);
         end;
    end;
    read_count_value32:=buffer.field0;
  END;
```

Fonction en „BORLAND C"

```
long read_count_value32 (unsigned int baseadr,
                  unsigned char axis, unsigned char reg)
{
union  mapper
        {
        long field0;
        unsigned int field1[2];
        }buffer;

switch (reg)
       {
       case 0:
       buffer.field1[0] = read_g26 (baseadr, axis, 0);
       buffer.field1[1] = read_g26 (baseadr, axis, 2);
       break;
       case 1:
       buffer.field1[0] = read_g26 (baseadr, axis, 6);
       buffer.field1[1] = read_g26 (baseadr, axis, 8);
       break;
       }
return (buffer.field0);
}
```

Fonction de lecture d'une valeur de mesure 48-bits
La fonction suivante permet de lire dans un compteur une valeu r de mesure 48□bits.

Fonction: read_count_value48

Paramètres

baseadr:	Adresse de base de l'IK 121 configurée avec les commutateurs DIP
axis:	1 = axe 1, 2 = axe 2
reg:	0 = registre de données 0 1 = registre de données 1
résultat:	„TURBO PASCAL": Valeur du registre de données indiquée sous **axis** et **reg** comme variable entière de type **comp** „BORLAND C": comme variable cirgule flottante de type **double**

Fonction en „TURBO PASCAL"

```
FUNCTION
 read_count_value48(baseadr:word;axis,reg:byte):comp;
  TYPE
    vartype = (li,by);
    mapper = record
               case wert:vartype of
                 li : (field0:comp);
                 by : (field1:array[0..3] of word);
               end;
  VAR
    buffer : mapper;
  BEGIN
    case reg of
     0 : begin
           buffer.field1[0]:=read_g26(baseadr,axis,0);
           buffer.field1[1]:=read_g26(baseadr,axis,2);
           buffer.field1[2]:=read_g26(baseadr,axis,4);
           if (buffer.field1[2] and $8000)=$8000 then
           buffer.field1[3]:=
                          $FFFF else buffer.field1[3]:=0;
         end;
     1 : begin
           buffer.field1[0]:=read_g26(baseadr,axis,6);
           buffer.field1[1]:=read_g26(baseadr,axis,8);
           buffer.field1[2]:=read_g26(baseadr,axis,10);
           if (buffer.field1[2] and $8000)=$8000 then
           buffer.field1[3]:=
                          $FFFF else buffer.field1[3]:=0;
         end;
     end;
     read_count_value48:=buffer.field0;
  END;
```

Fonction en „BORLAND C"

```
double read_count_value48 (unsigned int baseadr,
                  unsigned char axis, unsigned char reg)
{
unsigned int field[3];
double count_value48;

switch (reg)
       {
       case 0:
       field[0] = read_g26 (baseadr, axis, 0);
       field[1] = read_g26 (baseadr, axis, 2);
       field[2] = read_g26 (baseadr, axis, 4);
       break;
       case 1:
       field[0] = read_g26 (baseadr, axis, 6);
       field[1] = read_g26 (baseadr, axis, 8);
       field[2] = read_g26 (baseadr, axis, 10);
       break;
       }

if (field[2] && 0x8000)
       count_value48 = (double)((field[0]-65535) +
       65536.0*(field[1]-65535)+
       4294967296.0*(field[2]-65535)-1);
       else
       count_value48 = (double)(field[0] +
       65536.0*field[1] +
       4294967296.0*field[2]);


return (count_value48);
}
```

Programme simplifié pour appeler la valeur de mesure par le logiciel

Les fonctions définies précédemment sont utilisées dans les exemples de programmation qui suivent. Vous trouverez ces exemples sur la disquette 1 contenue dans la fourniture.
Pour „TURBO PASCAL" sous le répertoire **TP** dans les fichiers **SAMPLE32.PAS** et **SAMPLE48.PAS**.
Pour „BORLAND C" sous le répertoire **BC** dans les fichiers **SAMPLE32.C** et **SAMPLE48.C**.
Dans cet exemple, la position du compteur est affichée à l'écran en millimètres. Bien entendu, l'IK 121 peut aussi afficher des degrés angulaires. Les deux formules suivantes indiquent comment s'effectue la conversion.

Conversion de la position du compteur en millimètres

$$\text{Valeur[mm]} = \text{position du compteur} \cdot \frac{\text{Période de division[mm]}}{1024}$$

Exemple: Période de division = 20 µm

$$\text{Valeur[mm]} = \text{position du compteur} \cdot \frac{0{,}020\text{[mm]}}{1024}$$

Conversion de la position du compteur en degrés

$$\text{Valeur[°]} = \frac{\text{Position compteur} \cdot 360\text{[°]}}{1024 \cdot \text{traits/tour}}$$

Exemple: Nombre de traits/tour = 36 000

$$\text{Valeur[°]} = \frac{\text{Position compteur} \cdot 360\text{[°]}}{1024 \cdot 36000}$$

Exemples en „TURBO PASCAL": „Appel de la valeur de me sure par le logiciel "

Les exemples suivants SAMPLE32.PAS et SAMPLE48.PAS illustre nt l'utilisation des procédures et des fonctions décrites précédemm ent. Ils sont contenus et définis dans le fichier **IK121_0.TPU** (tous l es fichiers se trouvent sur la disquette 1 contenue dans la fournitur e).

Avant de compiler SAMPLE0.PAS, il convient de compiler le fichier IK121_0.PAS vers l'„unité" IK121_0.TPU.

Dans la mesure où cet exemple simple ne lit pas l'adresse de l'IK□1 21 à partir du fichier IK 121.INI, cette adresse est définie comme co nstante „base_address".

```
program sample32;
{-------------------------------------------------------
 DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany
 Programme simplifié destiné à l'IK 121 pour la visualisation
 de deux axes. Valeur de mesure sur 32 bits.
 V 1.01
 Avril 1995
 -------------------------------------------------------}
{$N+,E+}
{$V+}
{$R+}
USES crt,ik121_0;
CONST
  base_address = $330;
VAR
  c_value_0, c_value_1  : comp;
BEGIN
clrscr;
   (*Initialisation carte en mode avec interpolation, axe 1*)
write_g26 (base_address, 1, $0c, $0001);
   (*Initialisation carte en mode avec interpolation, axe 2*)
write_g26 (base_address, 2, $0c, $0001);
   (*Effacer l'erreur, lancer le compteur, axe 1*)
write_g26 (base_address, 1, $0e, $0048);
   (*Effacer l'erreur, lancer le compteur, axe 2*)
write_g26 (base_address, 2, $0e, $0048);
   (*Charger le registre de contrôle 2, axe 1*)
write_g26 (base_address, 1, $1c, $0008);
   (*Charger le registre de contrôle 2, axe 2*)
write_g26 (base_address, 2, $1c, $0008);
REPEAT
   (*Appel du logiciel dans le registre 0, axe 1*)
soft_l0 (base_address, 1);
   (*Appel du logiciel dans le registre 0, axe 2*)
soft_l0 (base_address, 2);
   (*Valeur de mesure de l'axe 1 mémorisée?*)
poll_latch (base_address, 1, 0);
   (*Lire la valeur de mesure, axe 1*)
c_value_0:= read_count_value32 (base_address, 1, 0);
   (*Valeur de mesure de l'axe 2 mémorisée?*)
poll_latch (base_address, 2, 0);
 (*Lire la valeur de mesure, axe 2*)
```

```
c_value_1:= read_count_value32 (base_address, 2, 0);
(*Afficher à l'écran la valeur de mesure*)
gotoxy(1,10);
write(c_value_0*0.02/1024:16:4,
       c_value_1*0.02/1024:16:4);
UNTIL KEYPRESSED;
END.
```

```
program sample48;
{-------------------------------------------------------
 DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany
 Programme simplifié destiné à l'IK 121 pour la visualisation
 de deux axes. Valeur de mesure sur 48 bits.
 V 1.01
 Avril 1995
 -------------------------------------------------------}
{$N+,E+}
{$V+}
{$R+}
USES crt,ik121_0;
CONST
  base_address = $330;
VAR
  c_value_0, c_value_1  : comp;
BEGIN
clrscr;
   (*Initialisation carte en mode avec interpolation, axe 1*)
write_g26 (base_address, 1, $0c, $0041);
   (*Initialisation carte en mode avec interpolation, axe 2*)
write_g26 (base_address, 2, $0c, $0041);
   (*Effacer l'erreur, lancer le compteur, axe 1*)
write_g26 (base_address, 1, $0e, $0048);
   (*Effacer l'erreur, lancer le compteur, axe 2*)
write_g26 (base_address, 2, $0e, $0048);
   (*Charger le registre de contrôle 2, axe 1*)
write_g26 (base_address, 1, $1c, $0008);
   (*Charger le registre de contrôle 2, axe 2*)
write_g26 (base_address, 2, $1c, $0008);
REPEAT
   (*Appel du logiciel dans le registre 0, axe 1*)
soft_l0 (base_address, 1);
   (*Appel du logiciel dans le registre 0, axe 2*)
soft_l0 (base_address, 2);
   (*Valeur de mesure de l'axe 1 mémorisée?*)
poll_latch (base_address, 1, 0);
   (*Lire la valeur de mesure, axe 1*)
c_value_0:= read_count_value48 (base_address, 1, 0);
   (*Valeur de mesure de l'axe 2 mémorisée?*)
poll_latch (base_address, 2, 0);
 (*Lire la valeur de mesure, axe 2*)
c_value_1:= read_count_value48 (base_address, 2, 0);
```

```
(*Afficher à l'écran la valeur de mesure*)
gotoxy(1,10);
write(c_value_0*0.02/1024:16:4,
       c_value_1*0.02/1024:16:4);
UNTIL KEYPRESSED;
END.
```

Exemples en „BORLAND C": „Appel de la valeur de mesure par le logiciel "

Les exemples suivants **SAMPLE32.C** et **SAMPLE48.C** illustrent l 'utilisation des fonctions décrites précédemment. Ils sont contenus et définis dans les fichiers IK121_0.H et IK121_0.C (tous les fichie rs sont sur la disquette 1 contenue dans la fourniture). Pour la com pilation, un projet doit être créé avec IK121_0.C et SAMPLE32. C ou SAMPLE48.C. Dans la mesure où cet exemple simple ne lit pas l'adresse de l'IK 121 à partir du fichier IK 121.INI, cette adresse est définie comme constante „base_address".

```
/*----------------------SAMPLE32.C----------------------
 DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany
 Programme simplifié destiné à l'IK 121 pour la visualisation
 de deux axes. Valeur de mesure sur 32 bits.
 V 1.01
 Avril 1995
 Fichiers-projet:      IK121_0.C, SAMPLE32.C
 Inclure fichier:      IK121_0.H
 ------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
#include „ik121_0.h"
#define base_address 0x0330
int main()
{
double c_value_0, c_value_1;
cls;
   /*Initialisation carte en mode avec interpolation, axe 1*/
write_g26 (base_address, 1, 0x0c, 0x0001);
   /*Initialisation carte en mode avec interpolation, axe 2*/
write_g26 (base_address, 2, 0x0c, 0x0001);
   /*Effacer l'erreur, lancer le compteur, axe 1*/
write_g26 (base_address, 1, 0x0e, 0x0048);
   /*Effacer l'erreur, lancer le compteur, axe 2*/
write_g26 (base_address, 2, 0x0e, 0x0048);
   /*Charger le registre de contrôle 2, axe 1*/
write_g26 (base_address, 1, 0x1c, 0x0008);
   /*Charger le registre de contrôle 2, axe 2*/
write_g26 (base_address, 2, 0x1c, 0x0008);
   /*Arrêt curseur*/
_setcursortype (_NOCURSOR);
while(!kbhit())
       {
            /*Appel du logiciel dans le registre 0, axe 1*/
```

```
       soft_l0 (base_address, 1);
           /*Appel du logiciel dans le registre 0, axe 2*/
       soft_l0 (base_address, 2);
           /*Valeur de mesure de l'axe 1 mémorisée?*/
       poll_latch (base_address, 1, 0);
          /*Lire la valeur de mesure, axe 1*/
       c_value_0 = (double)read_count_value32
                                  (base_address, 1, 0);
           /*Valeur de mesure de l'axe 2 mémorisée?*/
       poll_latch (base_address, 2, 0);
          /*Lire la valeur de mesure, axe 2*/
       c_value_1 = (double)read_count_value32
                                  (base_address, 2, 0);
        /*Afficher à l'écran la valeur de mesure*/
       printf(„\r\t%16.4f\t%16.4f",c_value_0*0.02/1024,
                                c_value_1*0.02/1024);
       }
   /*Rebrancher le curseur*/
_setcursortype (_NORMALCURSOR);
return (0);
}
/*----------------------SAMPLE48.C-----------------------
 DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany
 Programme simplifié destiné à l'IK 121 pour la visualisation
 de deux axes. Valeur de mesure sur 48 bits.
 V 1.01
 Avril 1995
 Fichiers-projet:      IK121_0.C, SAMPLE48.C
 Inclure fichier:      IK121_0.H
 -------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
#include „ik121_0.h"
#define base_address 0x0330
int main()
{
double c_value_0, c_value_1;
cls;
   /*Initialisation carte en mode avec interpolation, axe 1*/
write_g26 (base_address, 1, 0x0c, 0x0041);
  /*Initialisation carte en mode avec interpolation, axe 2*/
write_g26 (base_address, 2, 0x0c, 0x0041);
   /*Effacer l'erreur, lancer le compteur, axe 1*/
write_g26 (base_address, 1, 0x0e, 0x0048);
   /*Effacer l'erreur, lancer le compteur, axe 2*/
write_g26 (base_address, 2, 0x0e, 0x0048);
   /*Charger le registre de contrôle 2, axe 1*/
write_g26 (base_address, 1, 0x1c, 0x0008);
   /*Charger le registre de contrôle 2, axe 2*/
write_g26 (base_address, 2, 0x1c, 0x0008);
   /*Arrêt curseur*/
_setcursortype (_NOCURSOR);
while(!kbhit())
       {

/*Appel du logiciel dans le registre 0, axe 1*/
       soft_l0 (base_address, 1);
```

```
            /*Appel du logiciel dans le registre 0, axe 2*/
        soft_l0 (base_address, 2);
            /*Valeur de mesure de l'axe 1 mémorisée?*/
        poll_latch (base_address, 1, 0);
           /*Lire la valeur de mesure, axe 1*/
        c_value_0 = read_count_value48 (base_address, 1, 0);
            /*Valeur de mesure de l'axe 2 mémorisée?*/
        poll_latch (base_address, 2, 0);
           /*Lire la valeur de mesure, axe 2*/
        c_value_1 = read_count_value48 (base_address, 2, 0);
        /*Afficher à l'écran la valeur de mesure */
        printf(„\r\t%16.4f\t%16.4f",c_value_0*0.02/1024,
                                    c_value_1*0.02/1024);
        }
    /*Afficher le curseur*/
_setcursortype (_NORMALCURSOR);
return (0);
}
```

Fichier IK121_1.PAS: Fonctions pour modèle de mémoire RAM en „TURBO PASCAL"

Les structures de données et fonctions pour „TURBO PASCAL" décrites ci-après sont situées sur la disquette 1 contenue dans la fourniture, sous le répertoire **TP** dans les fichiers IK121_1.PAS et IIC.PAS (ce fichier est intégré à IK121_1.PAS lors de la compilation). Les fichiers sont intégrés comme **unité** à l'aide de l'instruction **uses** aux autres programmes d'applications. Dans le fichier IK121_1.PAS, un modèle de mémoire RAM des registres de l'IK□121 est créé à l'aide de structures de données (en „TURBO PASCAL": record).

Les données du modèle de mémoire RAM sont écrites dans les adresses de port de l'IK□121 à l'aide de **init_handler** et **comm_handler**.

Par ailleurs, on dispose de fonctions utiles permettant d'initialiser la carte, de lire les valeurs de mesure, de programmer les interruptions et d'aligner les signaux des systèmes de mesure.

Définition des structures de données

Record: softcommand

Cette structure est la reproduction de mémoire RAM du registre de contrôle 1 (accès à l'écriture sur 0Eh).

Champs de données

start:	Lancer le compteur
stop:	Stopper le compteur
clear:	Effacer le compteur
latch0:	Appel logiciel: valeur de mesure dans le registre de données 0
latch1:	Appel logiciel: valeur de mesure dans le registre de données 1
latch2:	Appel logiciel avec fonctions spéciales
clrfrg:	Effacer défaut du système de mesure (dépassement de fréquence)
clrstat:	Effacer le registre de valeur d'amplitude

Record: refcommand

Cette structure est la reproduction de mémoire RAM du registr e des marques de référence (accès à l'écriture sur 10h). Au fran chissement de la marque de référence, les fonctions suivantes peuvent être exécutées.

Champs de données

ristart: Lancer le compteur
ristop: Stopper le compteur
riclear: Effacer le compteur
riclear2: Effacer le compteur au franchissement de chaque marque de référence
rilatch: Appeler la valeur de mesure
rilatch2: Appeler la valeur de mesure au franchissement de la 2ème marque de référence

Record: initlatch

Cette structure est la reproduction de mémoire RAM du registr e de l'appel de la valeur de mesure (accès à l'écriture sur 12h).

Champs de données

en_L0_reg0: Validation L0 pour le registre de données 0
en_latch2_reg0: Validation de l'„appel de logiciel dans tous les registres de données" pour le registre de données 0
en_timer_reg0: Validation de l'appel de logiciel par timers pour le registre de données 0
en_L1_reg1: Validation L1 pour le registre de données 1
en_latch2_reg1: Validation de l'„appel de logiciel dans tous les registres de données" pour le registre de données 1
en_timer_reg1: Validation de l'appel de logiciel par timers pour le registre de données 1

Record: initsync

Cette structure est le reproduction de mémoire RAM du registr e de validation des axes en cascade (accès à l'écriture sur 13h). Les champs de données suivants s'appliquent uniquement à l'a xe 1.

Champs de données

en_L0_axis2: Validation du signal d'appel externe X3.L0 pour le 2nd axe
en_latch2_axis2: Validation de l'appel de logiciel avec fonctions spéciales pour le 2ème axe
en_timer_axis2: Validation du strobe de timer pour le 2ème axe.

Record: initmain

Cette structure est la reproduction de mémoire RAM des registres d'initialisation 1 et 2 (accès à l'écriture sur 0Ch et 0Dh).

Champs de données

mode1024: 0 = Mode compteur de périodes (sans interpolation, les bits de données D0 à D9 ne sont pas définis).
1 = Valeur de mesure créée à partir de la valeur du compteur de périodes et de la valeur d'interpolation.

en_timer: 0 = Remise à zéro et arrêt du timer
1 = Lancement du timer

en_48Bit: 0 = Mode: compteur 32 bits
1 = Mode: compteur 48 bits

onefold: Compteurs comptent 1 front/période signal

twofold: Compteurs comptent 2 fronts/période signal

fourfold: Compteurs comptent 4 fronts/période signal

arcmode: 0 = Comptage linéaire
1 = Comptage angulaire

arc180000: 0 = Comptage angulaire 17 999 à –18 000
1 = Comptage angulaire 179 999 à –180 000

sel_ri_1st: 0 = 1ère marque de référence mémorisée dans le registre de données 0
1 = 1ère marque de référence mémorisée dans le registre de données 1

sel_ri_2nd: 0 = 2ème marque de référence mémorisée dans le registre de données 0
1 = 2ème marque de référence mémorisée dans le registre de données 1

Record: initintrpt

Cette structure est la reproduction de mémoire RAM du registr e de validation interruptions (accès à l'écriture sur 14h).

Champs de données

reg0: Validation de l'interruption 0 pour appel de la valeur de mesure par registre de données 0

reg1: Validation de l'interruption 1 pour appel de la valeur de mesure par registre de données 1

timer: Validation de l'interruption 2 pour le strobe de timer

port: Interruption délivrée

Record: **intstate**

Cette structure est la reproduction de mémoire RAM des registres d'état interruptions 1 et 2 (accès lecture sur 14h et 15h).

Champs de données

L0:	Interruption 0 active, il y a eu appel de la valeur de mesure par registre de données 0
L1:	Interruption 1 active, il y a eu appel de la valeur de mesure par registre de données 1
timer:	Interruption 2 active, il y a eu appel de la valeur de mesure par strobe de timer
pendl0:	Interruption 0 présente mais n'est pas encore exécutée
pendl1:	Interruption 1 présente mais n'est pas encore exécutée
pendtimer:	Interruption 2 présente mais n'est pas encore exécutée

Record: **countstate**

Cette structure est la reproduction de mémoire RAM des regist res d'état 1 et 2 (accès à la lecture sur 0Eh et 0Fh).

Champs de données

latch0:	La valeur de mesure dans registre de données 0 est prête
latch1:	La valeur de mesure dans registre de données 1 est prête
stop:	Le compteur est stoppé
sda:	Ligne SDA du bus PC (entrée)
error_frq:	Défaut système de mesure (dépassement de fréquence)
ref_activ:	Le franchissement de la marque de référence est actif

Record: **signalstate**

Cette structure est la reproduction de mémoire RAM du registr e de valeur d'amplitude (accès à la lecture sur 11h), du registre d'amplitude pour le signal 0° (accès à la lecture sur 16h et 17 h) ainsi que du registre d'amplitude pour le signal 90° (accès à la le cture sur 18h et 19h).

Champs de données

ad00:	Amplitude pour le signal 0°
ad90:	Amplitude pour le signal 90°
amp_act:	Amplitude actuelle
amp_min:	Valeur min. de l'amplitude

Pointeur: **g26_pointr**

Pointeur sur une structure **g26_record**

Record: **g26_record**
Cette structure contient la séquence de données complète de l a reproduction de mémoire RAM des registres d'un axe.

Pointeur: **ik121_pointr**
Pointeur sur une structure ik121_record

Record: **ik121_record**
Zone des pointeurs **g26_pointr**

Record: **storage**
Zone de structures **g26_record** pour le fichier IK121.INI destiné à la mémorisation des valeurs d'initialisation

Procédures et fonctions
Fonction: **look_for_IK121**
Cette fonction permet de contrôler si le hardware de l'IK□121 e xiste.
Résultat: true, si l'IK 121 existe
false, si l'IK 121 n'existe pas
Prototype:
FUNCTION look_for_IK121 (board:IK121_pointr):boolean;

Procédure: **init_IK121**
Cette procédure initialise l'IK 121.
Prototype:
PROCEDURE init_ik121 (board:IK121_pointr);

Procédure: **init_handler**
Cette procédure écrit les données d'initialisation du modèle de mémoire RAM dans les adresses de port de l'IK 121 à l'aide de s fonctions suivantes: g26_main, g26_latch, g26_sync et g26_in t.
Prototype: PROCEDURE init_handler
(pointr:g26_pointr);

Procédure: **comm_handler**
Cette procédure écrit les instructions du modèle de mémoire R AM dans les adresses de ports de l'IK 121 à l'aide des fonction s suivantes: g26_soft, g26_ref.
Prototype: PROCEDURE comm_handler
(pointr:g26_pointr);

Procédure: **read_adr**
Cette procédure lit l'adresse de l'IK121 dans le fichier IK121.INI
Prototype: PROCEDURE read_adr (board:ik121_pointr);

Procédure: **write_adr**
Cette procédure écrit l'adresse de l'IK 121 dans le fichier IK121. INI
Prototype: PROCEDURE write_adr (board:ik121_pointr);

Procédure: **read_signal_status**
Cette procédure lit les données pour la structure **signalstate** da ns les registres 11h, 16h, 17h, 18h et 19h.
Prototype: PROCEDURE read_signal_status (pointr:g26_pointr);

Procédure: **read_count_status**
Cette procédure lit les données pour la structure **countstate** da ns les registres 0Eh et 0Fh.
Prototype: PROCEDURE read_count_status (pointr:g26_pointr);

Procédure: **read_int_status**
Cette procédure lit les données pour la structure **intstate** dans l es registres 14h et 15h.
Prototype: PROCEDURE read_int_status (pointr:g26_pointr);

Procédure: **soft_latch0**
Cette procédure mémorise une valeur de mesure dans le regist re de données 0.
Prototype: PROCEDURE soft_latch0 (pointr:g26_pointr);

Procédure: **soft_latch1**
Cette procédure mémorise une valeur de mesure dans le regist re de données 1.
Prototype: PROCEDURE soft_latch1 (pointr:g26_pointr);

Procédure: **read_reg0**
Cette procédure lit une valeur de mesure dans le registre de do nnées 0.
Prototype: PROCEDURE read_reg0 (pointr:g26_pointr);

Procédure: **read_reg1**
Cette procédure lit une valeur de mesure dans le registre de do nnées 1.
Prototype: PROCEDURE read_reg1 (pointr:g26_pointr);

Procédure: **poll_reg0**
Cette procédure interroge le bit d'état D0 dans le registre 0Eh j usqu'à ce qu'une valeur de mesure ait été mémorisée dans le r egistre de données 0.
Prototype: PROCEDURE poll_reg0 (pointr:g26_pointr);

Procédure: **poll_reg1**
Cette procédure interroge le bit d'état D1 dans le registre 0Eh j usqu'à ce qu'une valeur de mesure ait été mémorisée dans le r egistre de données 1.
Prototype: PROCEDURE poll_reg1 (pointr:g26_pointr);

Procédure: **en_int**
Cette procédure délivre une interruption.
Prototype: PROCEDURE en_int (Intrpt:byte);

Procédure: **dis_int**
Cette procédure bloque une interruption.
Prototype: PROCEDURE dis_int (Intrpt:byte);

Procédure: **clear_int**
Cette procédure annule une interruption.
Prototype: PROCEDURE clear_int;

Procédure: **write_offset**
Cette procédure inscrit des valeurs de correction d'offset dans l e registre d'offset des compteurs (pas de sauvegarde en cas de coupure d'alimentation!). Pour mémoriser les valeurs de correc tion d'offset à l'abri des coupures d'alimentation et les remémo riser dans les registres d'offset, utiliser les procédures „store_o ffset" und „load_offset".
Prototype: PROCEDURE write_offset
(baseadr:word;axis:byte;offx,offy:integer);

Les fonctions suivantes sont définies dans le fichier IIC.PAS . Après compilation vers une „unité", ce fichier est intégré dans IK121_1.PAS.
Procédure: **load_offset**
Cette procédure lit l'offset dans les registres d'offset.
Prototype: PROCEDURE load_offset
(board:IK121_pointr;var error:boolean);

Procédure: **store_offset**
Cette procédure mémorise une valeur de correction dans les re gistres d'offset.
Prototype: PROCEDURE store_offset
(board:IK121_pointr;var error:boolean)

Procédure: **poti_default**
Cette procédure met les potentiomètres à la position zéro.
Prototype: PROCEDURE poti_default
(pointr:ik121_pointr;var error:boolean);

Fonction: **read_phasepoti**
Cette fonction lit la position du potentiomètre pour la position d e phase.
Prototype: FUNCTION read_phasepoti
(pointr:ik121_pointr;axis:byte;var error:boolean):byte;

Fonction: **read_sympoti**
Cette fonction lit la position du potentiomètre pour la symétrie.
Prototype: FUNCTION read_sympoti
(pointr:ik121_pointr;axis:byte;var error:boolean):byte;

Procédure: **write_phasepoti**
Cette procédure règle le potentiomètre pour la position de phas e à une valeur prédéfinie.
Prototype: PROCEDURE write_phasepoti
(pointr:ik121_pointr;axis,wert:byte;var error:boolean);

Procédure: **write_sympoti**
Cette procédure règle le potentiomètre pour la symétrie à une v aleur prédéfinie.
Prototype: PROCEDURE write_sympoti
(pointr:ik121_pointr;axis,wert:byte;var error:boolean);

Procédure: **write_offset00**
Cette procédure écrit une valeur de correction dans le registre d 'offset pour le signal 0°.
Prototype: PROCEDURE write_offset00
(pointr:ik121_pointr;axis,value:integer);

Procédure: **write_offset90**
Cette procédure écrit une valeur de correction dans le registre d 'offset pour le signal 90°.
Prototype: PROCEDURE write_offset90
(pointr:ik121_pointr;axis,value:integer);

Procédure: **turn_phasepoti**
Cette procédure ajuste le potentiomètre pour la position de pha se.
Prototype: PROCEDURE turn_phasepoti
(pointr:ik121_pointr;axis,turns:byte;updown:boolean;var error:bo olean);

Procédure: **turn_sympoti**
Cette procédure ajuste le potentiomètre pour la symétrie.
Prototype: PROCEDURE turn_sympoti
(pointr:ik121_pointr;axis,turns:byte;updown:boolean;var error:bo
olean);

Procédure: **turn_offsetdg00**
Cette procédure modifie la valeur de correction du registre d'off
set pour le signal 0°.
Prototype: PROCEDURE turn_offsetdg00
(pointr:ik121_pointr;axis,turns:byte;updown:boolean)

Procédure: **turn_offsetdg90**
Cette procédure modifie la valeur de correction du registre d'off
set pour le signal 90°.
Prototype: PROCEDURE turn_offsetdg90
(pointr:ik121_pointr;axis,turns:byte;updown:boolean)

Procédure: **store_potis**
Cette procédure mémorise les positions du potentiomètre.
Prototype: PROCEDURE store_potis
(pointr:ik121_pointr;var error:boolean);

Procédure: **rom_write**
Cette procédure écrit à l'intérieur de la mémoire EEPROM libre.
Prototype: PROCEDURE rom_write
(pointr:ik121_pointr;adr;data :byte;var error : boolean);

Fonction: **rom_read**
Cette fonction lit la mémoire EEPROM libre.
Prototype: FUNCTION rom_read
(pointr:ik121_pointr;adr:byte;var error:boolean) :byte;

Programmes d'applications avec le modèle de mémoire RAM en „TURBO PASCAL"

Les fichiers EXE des exemples suivants qui peuvent être exécutés sont mémorisés dans le répertoire principal de la disquette 1 contenue dans la fourniture. Les fichiers source se trouvent dans le sous-répertoire **\TP**. Les programmes nécessitent l'utilisation d'un driver graphique BORLAND (BORLAND Gaphics Interface = *.BGI). La fourniture de l'IK□121 comprend le driver EGA VGA.BGI. Il convient de le mémoriser dans le même répertoire que celui des exemples de programmation.

SAMPLE1.EXE

L'exemple SAMPLE1.EXE est un programme d'application simplifié permettant d'afficher le contenu des registres de données 0 de l'axe 1 et de l'axe 2.

Code source:	SAMPLE1.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM

SAMPLE2.EXE

L'exemple SAMPLE2.EXE illustre l'utilisation de la programmation interruptions. IRQ14 est utilisé pour Int1, et IRQ15 pour Int 0. Une interruption est déclenchée par un front descendant sur X3.L0 ou X3.L1.

Code source:	SAMPLE2.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM

SAMPLE3.EXE

L'exemple SAMPLE3.EXE illustre l'utilisation de l'IK 121 pour mesurer la vitesse. La vitesse et l'accélération de l'axe 1 sont affichées. L'accélération est calculée à partir des vitesses ds(1) et ds(2).

Code source:	SAMPLE3.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM

SAMPLE4.EXE

L'exemple SAMPLE4.EXE montre la manière dont on peut lire l es potentiomètres électroniques pour la position de phase et l'a mplitude.

Code source:	SAMPLE4.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM

SAMPLE5.EXE

L'exemple SAMPLE5.EXE montre pour l'axe 1 le contenu des r egistre 0 et 1. Par ailleurs, pour le registre 0, le contenu du com pteur de périodes et la valeur d'interpolation sont affichés sépar ément. Vous pouvez introduire au clavier différentes command es explicitées à l'écran.

Code source:	SAMPLE5.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM

SAMPLE6.EXE

L'exemple SAMPLE6.EXE illustre l'utilisation de l'IK 121 en tant que compteur de périodes pour l'axe 1 et l'axe 2; ce qui signifi e que la valeur d'interpolation n'est pas exploitée. Vous pouvez introduire au clavier différentes commandes explicitées à l'écra n.

Code source:	SAMPLE6.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM

SCOPE.EXE

L'exemple SCOPE.EXE montre les signaux sinusoïdaux des sys tèmes de mesure raccordés soit sous forme d'un dia-gramme amplitude-durée, soit sous forme de représentation XY . Les potentiomètres peuvent être réglés à l'aide des softkeys d ont la signification est donnée à l'écran.

Code source:	SCOPE.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM
	IK121_2.TPU	Fonctions pour ADJUST.EXE, POTIS.EXE et SCOPE.EXE
	SCOPE_0.TPU	Fonctions pour SCOPE.EXE
	CNT_0.TPU	Fonctions fenêtre
	LOGO.TPU	Ecran d'accueil au lancement du programme

POTIS.EXE

L'exemple POTIS.EXE indique la position des potentiomètres él ectroniques pour la position de phase et l'amplitude ainsi que le s valeurs des registres d'offset. Les potentiomètres peuvent êtr e réglés à l'aide des softkeys dont la signification est donnée à l 'écran.

Code source:	POTIS.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM
	IK121_2.TPU	Fonctions pour ADJUST.EXE
	POTI_0.TPU	Fonctions pour POTIS.EXE
	CNT_0.TPU	Fonctions fenêtre
	LOGO.TPU	Ecran d'accueil au lancement du programme

ADJUST.EXE

L'exemple ADJUST.EXE exécute un alignement automatique d e l'axe 1 (sélection:1) ou de l'axe 2 (sélection:2) pour la position de phase (sélection:p), l'amplitude (sélection:a) et l'offset (sélec tion:o) des signaux sinusoïdaux des systèmes de mesure. Les v aleurs de compensation sont générées par un déplacement lent du système de mesure. A l'issue de 30 pé-
riodes de signal, un signal sonore indique qu'une valeur de com pensation a été générée et qu'elle peut être mémorisée en app uyant sur la touche S.

Code source:	ADJUST.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM
	IK121_2.TPU	Fonctions pour ADJUST.EXE, POTIS.EXE et SCOPE.EXE
	ADJ_0.TPU	Fonctions pour ADJUST.EXE
	CNT_0.TPU	Fonctions fenêtre
	LOGO.TPU	Ecran d'acceuil au lancement du programme

IK121.EXE

Le programme IK121.EXE est un exemple d'application pour un programme de visualisation de positions en „TURBO PASCAL".

Code source:	IK121.PAS	
Units:	IK121_0.TPU	Fonctions de base
	IK121_1.TPU	Fonctions pour un modèle de mémoire RAM
	IK121_2.TPU	Fonctions pour ADJUST.EXE, POTIS.EXE et SCOPE.EXE
	CNT_0.TPU	Fonctions fenêtre
	CNT_1.TPU	Affichage de positions
	CNT_2.TPU	Programme compteur
	SCOPE_0.TPU	Fonctions pour SCOPE.EXE
	ADJ_0.TPU	Fonctions pour ADJUST.EXE
	LOGO.TPU	Ecran d'accueil au lancement du programme
Include-	IK121.WIN	Définition fenêtre
Fichiers:	IK121-CNT	Fichier d'initialisation
Aide:	IK121.HLP	Fichier avec texte d'aide

En cliquant avec la souris ou à l'aide des touches fléchées horiz ontales, vous pouvez ouvrir un menu pull-down. Vous disposez des fonctions suivantes:

File	Modifier et mémoriser les configurations de la carte, des axes et de l'imprimante
Adjust	Afficher et aligner les signaux sinusoïdaux des systèmes de mesure
Counters	Initialiser les axes, ou les remettre à zéro, ou lancer le franchissement des marques de référence
Sorting	Mode de fonctionnement „classification": introduire une valeur limite basse et haute
Datum	Sélectionner le plan de référence. Quatre plans de référence (L0 à L3) sont disponibles pour les deux axes
MIN/MAX	Activer l'affichage des valeurs limites min. et max. d'un cycle de mesures
Print	Restituer valeur de mesure sur l'imprimante configurée sous „File/setupIK121/printer". Si vous avez sélectionné „File:on" comme imprimante, la valeur de mesure sera restituée dans le fichier IK121.DAT.

Les champs d'affichage près des affichages de positions ont la signification suivante:

signal	Défaut intervenu sur le système de mesure
Ref	Franchissement de la marque de référence a a été sélectionné
<	Valeur de mesure inférieure à la valeur limite basse introduite sous „Sorting"
>	Valeur de mesure supérieure à la valeur limite haute introduite sous „Sorting"

Si vous hésitez dans l'utilisation du programme IK 121, appuy ez simplement sur la touche F1: l'écran vous présente l'aide r elative à la fonction sélectionnée.

EEPROM vacante

La platine de l'IK 121 est équipée d'une EEPROM de mémoire 512 octets adressée par le bus I²C. Pour écrire et lire l'EEPROM , vous disposez des programmes suivants situés sur la disquett e contenue dans la fourniture:

RDROM.EXE
Le programme RDROM.EXE lit le contenu de la mémoire de l'E EPROM.

Code source:	RDROM.PAS	
Units:	IK 121_1.TPU	Fonctions pour un modèle de mémoire RAM

WRROM.EXE
Le programme WRROM.EXE écrit le fichier IK121.TXT à l'intéri eur de la mémoire de l'EEPROM.

Code source:	RDROM.PAS	
Units:	IK 121_1.TPU	Fonctions pour un modèle de mémoire RAM

Exemples d'applications avec le modèle de mémoire RAM en „BORLAND C++“

Les exemples avec le modèle de mémoire RAM en „BORLAN D C++“ sont contenus sur la disquette 1, dans le répertoire **BC PP**. Les structures de données et fonctions utilisées sont signal ées et définies dans les fichiers suivants:

IK121.H:	Dans ce fichier header, vous pouvez définir un e adresse pour les IK (jusqu'à 16 IK 121).
IK121_1.H:	Structures de données pour un modèle de mé moire RAM des registres de l'IK□121 et indica tion des fonctions dans les fichiers IK121_1.C PP, IIC.CPP et POTI_1.CPP
IK121_1.CPP:	Fonctions de base pour l'IK□121
IIC.CPP:	Fonctions de transfert de données via le bus I² C.
POTI_1.CPP:	Fonctions de réglage des potentiomètres élec troniques.

Un modèle de mémoire RAM des registres de l'IK□121 est bâti dans le fichier IK121_1.H à l'aide de structures de données. Les données de ce modèle de mémoire RAM sont écrites dans les registres de l'IK□121 grâce aux procédures **InitHandler** et **Com mHandler**.

POTIS.EXE
Le programme POTIS.EXE montre la manière dont vous pouvez régler par logiciel les potentiomètres électroniques de l'IK□121 par le bus I²C.
Code source: POTIS.CPP, IK121_1.CPP
IIC.CPP, POTI_1.CPP
Fichiers header: IK 121.H, IK121_1.H

RDROM.EXE
Le programme RDROM.EXE lit le contenu de la mémoire EEPROM.
Code source: RDROM.CPP, IK121_1.CPP, IIC.CPP
Fichiers header: IK121.H, IK 121_1.H

WRROM.EXE
Le programme WRROM.EXE écrit le fichier IK121.TXT dans la mémoire EEPROM.
Code source: WRROM.CPP, IK121_1.CPP, IIC.CPP
Fichiers header: IK121.H, IK 121_1.H

DISPLAY.EXE
Le programme DISPLAY.EXE affiche le contenu de tous les registres de l'IK□121.
Code source: DISPLAY.CPP, IK121_1.CPP, IIC.CPP
Fichiers header: IK121.H, IK 121_1.H

L'IK 121 dans les applications WINDOWS

Sur les disquettes 2 et 3 jointes à la fourniture, vous trouverez l e logiciel driver, les Dynamic Link Libraries (DLL) ainsi que des exemples d'applications en VISUAL C++, VISUAL BASIC et BO RLAND DELPHI pour WINDOWS NT et WINDOWS 95.

Disquette 2
La disquette 2 contient la structure de répertoires suivante:

Install	**Fichiers d'installation**
IK121Drv	**Driver pour WINDOWS NT**
Release	Version Release
IK121Dll	**Source Windows DLL**
Release	Version Release pour Windows NT
Release95	Version Release pour Windows 95
IK121Con	**Exemple pour utilisation sur console**
Release	Version Release exemple console
IK121App	**Exemple pour VISUAL C++**
Res	Ressources pour exemple Windows
Release	Version Release exemple Windows
IK121VB5	**Exemple pour VISUAL BASIC**
Doc	**Documentation**

Disquette 3
La disquette 3 contient des exemples pour BORLAND DELPHI.

Driver de périphérique pour Windows NT (IK121DRV.SYS)

Sur la disquette 2 et dans le répertoire **\IK121Drv**, vous trouver ez le driver de périphérique pour WINDOWS NT (version 3.51 e t 4.0) permettant l'accès à l'IK□121.

Pour que Windows NT puisse charger le driver, il faut qu'il ait ét é copié dans le répertoire système de Windows NT sous **\syste m32\drivers** (ex.: C:\WINNT\SYSTEM32\DRIVERS). Ceci achèv e le fichier Batch **Install.Bat.**

Introduction dans Registry

Le driver reçoit de la Registry l'information indiquant sur quelle adresse de port l'IK□121 est installée. Il est nécessaire d'introdu ire les données suivantes dans la Registry:

```
HKEY_LOCAL_MACHINE
  System
    CurrentControlSet
      Services
        IK121DRV
              ErrorControl        0x00000001
              Start               0x00000003
              Type                0x00000001
          Parameters
            IK_Base_10x00000330
            IK_Base_2    0x00000000
            IK_Base_3    0x00000000
            IK_Base_4    0x00000000
            IK_Base_5    0x00000000
            IK_Base_6    0x00000000
            IK_Base_7    0x00000000
            IK_Base_8    0x00000000
```

Sur la disquette 2 et dans le répertoire **\Install**, vous trouverez l es fichiers **IK121Drv.Reg** et **SetReg.Bat.** Le fichier **IK121Drv.R eg** contient les données indiquées ci-dessus que vous devez in scrire dans la Registry avec **Install.Bat**.

Dans le fichier **IK121Drv.Reg**, vous pouvez modifier l'adresse d e base de l'IK□121 et/ou ajouter d'autres adresses de base. Le driver gère jusqu'à huit IK□121. L'introduction dans la Registry d oit être actualisée avec **SetReg.Bat**.

Les DLL WindowsDLL (IK121Dll.Dll)

Ces DLL permettent aux exemples d'applications de répondre à l'IK□121. Il y a une DLL pour Windows NT et une autre pour Windows 95. Sous Windows NT, l'IK 121 est contactée au moyen du driver de périphérique pour Windows NT. La DLL pour Windows 95 accède directement aus registres de l'IK□121. Les fonctions fréquemment employées sont directement utilisables (start, stop, lecture de valeur de comptage, fonctions REF, etc.). Une programmation plus poussée de l'IK□121 est possible en accédant à ses registres (IKInputW, IKInputL, IKOutput etc.). Pour que les programmes d'applications puissent charger la DLL, le fichier **\IK121Dll\Release\IK121Dll.Dll** de la disquette 2 doit se trouver dans le répertoire système de Windows NT sous **System32** (ex.: **C:\Winnt\System32**). Avec Windows 95, le fichier **\IK121Dll\Release95\IK121Dll.Dll** de la disquette 2 doit être mémorisé dans le répertoire système sous **System** (ex.: **C:\Windows\System**). Le fichier Batch **Install.Bat** copie les fichiers dans le répertoire concerné.

Exemple pour l'utilisation d'une console

Vous trouverez une application console simple sur la disquette□2, dans le répertoire **\IK121Con\Release**.

Exemple pour VISUAL C++

Vous trouverez une application en VISUAL C++ sur la disquette□2, dans le répertoire **\IK121App\Release**.

Exemple pour VISUAL BASIC

Vous trouverez une application en VISUAL BASIC sur la disquette 2, dans le répertoire **\IK121VB5\Release**.

Exemple pour BORLAND DELPHI

Sur la disquette 3, vous trouverez une application en BORLAND DELPHI.

Installation du driver et des DLL sous WINDOWS□NT et WINDOWS 95

- ➤ Sélectionnez le répertoire **\Install** sur la disquette 2 contenue dans la fourniture.
- ➤ Dans le fichier **IK121Drv.Reg**, inscrivez les adresses de port de l'IK□121 installée.
- ➤ Appelez **Install.Bat**.

Install.bat génère les données dans la Registry, copie le driver pour WINDOWS NT à partir du répertoire **\IK121Drv\Release** vers le répertoire-système (ex.: **C:\Winnt\System32\Drivers**) ainsi que les DLL pour WINDOWS NT (ou WINDOWS 95) à partir du répertoire **\IK121Dll\Release** (ou \IK121Dll\Release95) vers le répertoire-système (ex.: **C:\Winnt\System32**).

Appel des fonctions DLL à partir de vos propres programmes utilisateur

Avoir de pouvoir utiliser les fonctions des DLL, vous devez tour d'abord annoncer le programme utilisateur.

MICROSOFT VISUAL C++

Lorsque vous créez le programme utilisateur avec VISUAL C++ , le fichier **\IK121Dll\Release\IK121Dll.Lib** de la disquette 2 doi t se trouver dans le répertoire de librairie de VISUAL C++ (ex.: **C:\Msdev\lib**) et être relié au projet. Pour cela, vous créez une entrée en VISUAL C++, sous **Build, Settings, Link, Object/Lib rary modules.**

Dans votre fichier header, vous devez en outre définir les protot ypes suivants:

```
#ifdef __cplusplus
  extern "C"
  {
#endif

WINUSERAPI BOOL WINAPI IKFind (ULONG* pBuffer8);
WINUSERAPI BOOL WINAPI IKInit (USHORT Axis, USHORT Mode);

WINUSERAPI BOOL WINAPI IKReset (USHORT Axis);
WINUSERAPI BOOL WINAPI IKStart (USHORT Axis);
WINUSERAPI BOOL WINAPI IKStop (USHORT Axis);
WINUSERAPI BOOL WINAPI IKLatch (USHORT Axis, USHORT Latch);

WINUSERAPI BOOL WINAPI IKResetREF (USHORT Axis);
WINUSERAPI BOOL WINAPI IKStartREF (USHORT Axis);
WINUSERAPI BOOL WINAPI IKStopREF (USHORT Axis);
WINUSERAPI BOOL WINAPI IKLatchREF (USHORT Axis, USHORT Latch);

WINUSERAPI BOOL WINAPI IKLatched (USHORT Axis, USHORT Latch, BOOL* pStatus);
WINUSERAPI BOOL WINAPI IKWaitLatch (USHORT Axis, USHORT Latch);

WINUSERAPI BOOL WINAPI IKStrtCodRef (USHORT Axis, USHORT Latch, ULONG RefDist);
WINUSERAPI BOOL WINAPI IKCodRef (USHORT Axis, BOOL* pStatus, double* pData);
WINUSERAPI BOOL WINAPI IKWaitCodRef (USHORT Axis, double* pData);
WINUSERAPI BOOL WINAPI IKStopCodRef (USHORT Axis);
```

```
WINUSERAPI BOOL WINAPI IKClear (USHORT Axis);
WINUSERAPI BOOL WINAPI IKStatus (USHORT Axis, ULONG* pStatus);

WINUSERAPI BOOL WINAPI IKRead32 (USHORT Axis, USHORT Latch, LONG* pData);
WINUSERAPI BOOL WINAPI IKRead48 (USHORT Axis, USHORT Latch, double* pData);

WINUSERAPI BOOL WINAPI IKReadPhase (USHORT Axis, BYTE* pData);
WINUSERAPI BOOL WINAPI IKWritePhase (USHORT Axis, BYTE   Data);
WINUSERAPI BOOL WINAPI IKLoadPhase (USHORT Axis, BYTE* pData);

WINUSERAPI BOOL WINAPI IKReadAmp (USHORT Axis, BYTE* pData);
WINUSERAPI BOOL WINAPI IKWriteAmp (USHORT Axis,  BYTE   Data);
WINUSERAPI BOOL WINAPI IKLoadAmp (USHORT Axis, BYTE* pData);

WINUSERAPI BOOL WINAPI IKReadOffset (USHORT Axis, SHORT* Ofs0, SHORT* Ofs90);
WINUSERAPI BOOL WINAPI IKWriteOffset (USHORT Axis, SHORT  Ofs0, SHORT  Ofs90);
WINUSERAPI BOOL WINAPI IKLoadOffset (USHORT Axis, SHORT* Ofs0, SHORT* Ofs90);

WINUSERAPI BOOL WINAPI IKStore (USHORT Axis);
WINUSERAPI BOOL WINAPI IKDefault (USHORT Axis);

WINUSERAPI BOOL WINAPI IKRomRead (USHORT Card, BYTE Adr, BYTE* Data);
WINUSERAPI BOOL WINAPI IKRomWrite (USHORT Card, BYTE Adr, BYTE  Data);

WINUSERAPI BOOL WINAPI IKInputW (USHORT Axis, USHORT Adr, USHORT* pData);
WINUSERAPI BOOL WINAPI IKInputL (USHORT Axis, USHORT Adr, ULONG*   pData);
WINUSERAPI BOOL WINAPI IKOutput (USHORT Axis, USHORT Adr, USHORT    Data);

WINUSERAPI BOOL WINAPI IKSetI2C (USHORT Card, BOOL SCL, BOOL SDA);
#ifdef __cplusplus
  }
endif
```

Vous pouvez ensuite utiliser les fonctions comme des fonctions C „normales“.

MICROSOFT VISUAL BASIC

En VISUAL BASIC, vous pouvez définir les fonctions dans un m odule de la manière suivante:

```
Public Declare Function IKFind          Lib "IK121DLL.DLL"
                                        (ByRef pBuffer8 As Long) As Boolean
Public Declare Function IKInit Lib "IK121DLL.DLL"
                                        (ByVal Axis As Integer, ByVal Mode As Integer) As Boole
                                        an
```

```
Public Declare Function IKReset Lib "IK121DLL.DLL"
(ByVal Axis As Integer) As Boolean
Public Declare Function IKStart Lib "IK121DLL.DLL"
(ByVal Axis As Integer) As Boolean
Public Declare Function IKStop Lib "IK121DLL.DLL"
(ByVal Axis As Integer) As Boolean
Public Declare Function IKLatch Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByVal Latch As Integer) As Boole
an

Public Declare Function IKResetREF Lib "IK121DLL.DLL"
(ByVal Axis As Integer) As Boolean
Public Declare Function IKStartREF Lib "IK121DLL.DLL"
(ByVal Axis As Integer) As Boolean
Public Declare Function IKStopREF Lib "IK121DLL.DLL"
(ByVal Axis As Integer) As Boolean
Public Declare Function IKLatchREF Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByVal Latch As Integer) As Boole
an

Public Declare Function IKLatched Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByVal Latch As Integer, ByRef pS
tatus As Boolean) As Boolean
Public Declare Function IKWaitLatch Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByVal Latch As Integer) As Boole
an

Public Declare Function IKStrtCodRef Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByVal Latch As Integer, ByVal Re
fDist As Long) As Boolean
Public Declare Function IKCodRef Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByRef pStatus As Boolean, ByRe
f pData As Double) As Boolean
Public Declare Function IKWaitCodRef Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByRef pData As Double) As Bool
ean
Public Declare Function IKStopCodRef Lib "IK121DLL.DLL"
(ByVal Axis As Integer) As Boolean

Public Declare Function IKClear Lib "IK121DLL.DLL"
(ByVal Axis As Integer) As Boolean
Public Declare Function IKStatus Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByRef pStatus As Long) As Boole
an

Public Declare Function IKRead32 Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByVal Latch As Integer, ByRef pD
ata As Long) As Boolean
Public Declare Function IKRead48 Lib "IK121DLL.DLL"
(ByVal Axis As Integer, ByVal Latch As Integer, ByRef pD
ata As Double) As Boolean
```

```
Public Declare Function IKReadPhase  Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByRef pData As Byte) As Boolea
                                     n
Public Declare Function IKWritePhase Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByVal Data As Byte) As Boolean
Public Declare Function IKLoadPhase  Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByRef pData As Byte) As Boolea
                                     n

Public Declare Function IKReadAmp    Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByRef pData As Byte) As Boolea
                                     n
Public Declare Function IKWriteAmp   Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByVal Data As Byte) As Boolean
Public Declare Function IKLoadAmp    Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByRef pData As Byte) As Boolea
                                     n

Public Declare Function IKReadOffset Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByRef Ofs0 As Integer, ByRef Of
                                     s90 As Integer) As Boolean
Public Declare Function IKWriteOffset Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByVal Ofs0 As Integer, ByVal Ofs
                                     90 As Integer) As Boolean
Public Declare Function IKLoadOffset Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByRef Ofs0 As Integer, ByRef Of
                                     s90 As Integer) As Boolean

Public Declare Function IKStore      Lib "IK121DLL.DLL" (ByVal Axis) As Boolean
Public Declare Function IKDefault    Lib "IK121DLL.DLL" (ByVal Axis) As Boolean

Public Declare Function IKRomRead    Lib "IK121DLL.DLL"
                                     (ByVal Card As Integer, ByVal Adr As Byte, ByRef Data A
                                     s Byte) As Boolean
Public Declare Function IKRomWrite   Lib "IK121DLL.DLL"

                                              (ByVal Card As Integer, ByVal Adr As Byte, By
                                     Val Data As Byte) As Boolean

Public Declare Function IKInputW     Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByVal Adr As Integer, ByRef pDat
                                     a As Long) As Boolean
Public Declare Function IKInputL     Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByVal Adr As Integer, ByRef pDat
                                     a As Long) As Boolean
Public Declare Function IKOutput     Lib "IK121DLL.DLL"
                                     (ByVal Axis As Integer, ByVal Adr As Integer, ByVal Data
                                     As Long) As Boolean

Public Declare Function IKSetI2C     Lib "IK121DLL.DLL"
                                     (ByVal Card As Integer, ByVal SCL As Boolean, ByVal SD
                                     A As Boolean) As Boolean
```

BORLAND DELPHI

En BORLAND DELPHI, vous reliez les fonctions dans votre pro gramme de la manière suivante:

```
Function IKFind          (pBuffer8: Long8Ptr) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKInit          (Axis: Word; Mode: Word) : Boolean; StdCall; External
                         'IK121DLL.DLL';

Function IKReset         (Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKStart         (Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKStop          (Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKLatch         (Axis: Word; Latch: Word) : Boolean; StdCall; External
                         'IK121DLL.DLL';

Function IKResetREF      (Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKStartREF      (Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKStopREF       (Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKLatchREF      (Axis: Word; Latch: Word) : Boolean; StdCall; External
                         'IK121DLL.DLL';

Function IKLatched (Axis: Word; Latch: Word; pStatus: BoolPtr) : Boolean; StdCall;
                         External 'IK121DLL.DLL';
Function IKWaitLatch     (Axis: Word; Latch: Word) : Boolean; StdCall; External
                         'IK121DLL.DLL';

Function IKStrtCodRef    (Axis: Word; Latch: Word; RefDist: LongInt) : Boolean; StdCall;
                         External 'IK121DLL.DLL';
Function IKCodRef  (Axis: Word; pStatus: BoolPtr; pData: DoublePtr) : Boolean;
                         StdCall; External 'IK121DLL.DLL';
Function IKWaitCodRef    (Axis: Word; pData: DoublePtr) : Boolean; StdCall; External
                         'IK121DLL.DLL';
Function IKStopCodRef    (Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';

Function IKClear         (Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKStatus        (Axis: Word; pStatus: LongPtr) : Boolean; StdCall; External
                         'IK121DLL.DLL';

Function IKRead32        (Axis: Word; Latch: Word; pData: LongPtr) : Boolean; StdCall;
                         External 'IK121DLL.DLL';
Function IKRead48        (Axis: Word; Latch: Word; pData: DoublePtr) : Boolean; StdCall;
                         External 'IK121DLL.DLL';

Function IKReadPhase     (Axis: Word; pData: BytePtr) : Boolean; StdCall; External
                         'IK121DLL.DLL';
Function IKWritePhase    (Axis: Word;   Data: Byte) : Boolean; StdCall; External
                         'IK121DLL.DLL';
Function IKLoadPhase     (Axis: Word; pData: BytePtr) : Boolean; StdCall; External
                         'IK121DLL.DLL';
```

Function IKReadAmp	(Axis: Word; pData: BytePtr) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKWriteAmp	(Axis: Word; Data: Byte) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKLoadAmp	(Axis: Word; pData: BytePtr) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKReadOffset	(Axis: Word; Ofs0: IntPtr; Ofs90: IntPtr) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKWriteOffset	(Axis: Word; Ofs0: Integer; Ofs90: Integer) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKLoadOffset	(Axis: Word; Ofs0: IntPtr; Ofs90: IntPtr) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKStore	(Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKDefault	(Axis: Word) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKRomRead	(Card: Word; Adr: Byte; Data: BytePtr) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKRomWrite	(Card: Word; Adr: Byte; Data: Byte) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKInputW	(Axis: Word; Adr: Word; pData: WordPtr) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKInputL	(Axis: Word; Adr: Word; pData: LongPtr) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKOutput	(Axis: Word; Adr: Word; Data: LongInt) : Boolean; StdCall; External 'IK121DLL.DLL';
Function IKSetI2C	(Card: Word; SCL: Boolean; SDA: Boolean) : Boolean; StdCall; External 'IK121DLL.DLL';

Sommaire des fonctions DLL

Fonction	Référence abrégée
Constater installation IK 121	BOOL IKFind (ULONG* pBuffer8);
Initialiser l'IK□121	BOOL IKInit (USHORT Axis, USHORT Mode);
Effacer le compteur	BOOL IKReset (USHORT Axis);
Lancer le compteur	BOOL IKStart (USHORT Axis);
Stopper le compteur	BOOL IKStop (USHORT Axis);
Mémoriser val. comptage	BOOL IKLatch (USHORT Axis, USHORT Latch);
Effacer compteur avec marq ue de réf. suivante	BOOL IKResetREF (USHORT Axis);
Lancer compteur avec marq ue de réf. suivante	BOOL IKStartREF (USHORT Axis);
Stopper compteur avec mar que de réf. suivante	BOOL IKStopREF (USHORT Axis);
Mémoriser valeur compteur avec marque de réf. suivant e	BOOL IKLatchREF (USHORT Axis, USHORT Latch);

Fonction	Référence abrégée
Interroger si valeur compteur mémorisée	BOOL IKLatched (USHORT Axis, USHORT Latch, BOOL* pStatus);
Attendre jusqu'à ce que la valeur du compteur soit mémorisée	BOOL IKWaitLatch (USHORT Axis, USHORT Latch);
Lance franchissement des marques de référence à distances codées	BOOL IKStrtCodRef (USHORT Axis, USHORT Latch, ULONG RefDist);
Interroger si franchissement point de référence achevé (marques de référence à distances codées)	BOOL IKCodRef (USHORT Axis, BOOL* pStatus, double* pData);
Attendre franchissement point de référence (marques de référence à distances codées)	BOOL IKWaitCodRef (USHORT Axis, double* pData);
Arrêter franchissement point de référence (marques de référence à distances codées)	BOOL IKStopCodRef (USHORT Axis);
Effacer erreurs de fréquence et d'ampliltude	BOOL IKClear (USHORT Axis);
Interrogation sur l'état	BOOL IKStatus (USHORT Axis, ULONG* pStatus);
Lire la valeur de compteur mémorisée (32 bits)	BOOL IKRead32 (USHORT Axis, USHORT Latch, LONG* pData);
Lire la valeur de compteur mémorisée (48 bits)	BOOL IKRead48 (USHORT Axis, USHORT Latch, double* pData);
Lire la valeur de correction de phase	BOOL IKReadPhase (USHORT Axis, BYTE* pData);
Modifier la valeur de correction de phase	BOOL IKWritePhase (USHORT Axis, BYTE Data);
Lire la valeur de correction de phase mémorisée	BOOL IKLoadPhase (USHORT Axis, BYTE* pData);
Lire la valeur de correction d'amplitude	BOOL IKReadAmp (USHORT Axis, BYTE* pData);
Modifier la valeur de correction d'amplitude	BOOL IKWriteAmp (USHORT Axis, BYTE Data);

Fonction	Référence abrégée
Lire la valeur de correction d' amplitude mémorisée	BOOL IKLoadAmp (USHORT Axis, BYTE* pData);
Lire la valeur de correction d' offset	BOOL IKReadOffset (USHORT Axis, SHORT* Ofs0, SHORT* Ofs 90);
Modifier la valeur de correcti on d'offset	BOOL IKWriteOffset (USHORT Axis, SHORT Ofs0, SHORT Ofs 90);
Lire la valeur de correction d' offset mémorisée	BOOL IKLoadOffset (USHORT Axis, SHORT* Ofs0, SHORT* Ofs 90);
Mémoriser val. correction	BOOL IKStore (USHORT Axis);
Chager et mémoriser les val eurs neutres	BOOL IKDefault (USHORT Axis);
Lire valeur dans EEPROM	BOOL IKRomRead (USHORT Card, BYTE Adr, BYTE* Data);
Ecrire valeur dans EEPROM	BOOL IKRomWrite (USHORT Card, BYTE Adr, BYTE Data);
Lire registre IK121 (16 bits)	BOOL IKInputW (USHORT Axis, USHORT Adr, USHORT* pData);
Lire registre IK121 (32 bits)	BOOL IKInputL (USHORT Axis, USHORT Adr, ULONG* pData);
Ecrire registre IK121 (16bits)	BOOL IKOutput (USHORT Axis, USHORT Adr, USHORT Data);
Initialiser lignes I^2C	BOOL IKSetI2C (USHORT Card, BOOL SCL, BOOL SDA);

Référence des fonctions DLL

Toutes les fonctions DLL délivrent en retour une variable Bool:
true (<> 0) si la fonction a été exécutée avec succès et
false (= 0) si une erreur s'est produite.

IKFind

Cette fonction délivre les adresses de port de l'IK□121 installée.
Les entrées non utilisées sont mises à 0.

Prototype: BOOL IKFind (ULONG* pBuffer[8]);

pBuffer: pointeur sur 8 mots longs (4 bytes)

IKInit

Cette fonction initialise l'IK□121.

Prototype: BOOL IKInit (USHORT Axis, USHORT Mode);

Axis: Numéro de l'axe (0 à 15)

Mode: 0=valeur compteur 32 bits
1= valeur compteur 48 bits

IKReset

Cette fonction met le compteur à zéro.

Prototype: BOOL IKReset (USHORT Axis);

Axis: Numéro de l'axe (0 à 15)

IKStart
Cette fonction lance le compteur.
Prototype: BOOL IKStart (USHORT Axis);
Axis: Numéro de l'axe (0 à 15)

IKStop
Cette fonction arrête le compteur.
Prototype: BOOL IKStop (USHORT Axis);
Axis: Numéro de l'axe (0 à 15)

IKLatch
Cette fonction mémorise la valeur du compteur.
Prototype: BOOL IKLatch (USHORT Axis, USHORT Latch);
Axis: Numéro de l'axe (0 à 15)
Latch: 0=valeur compteur mémorisée dans le registre 0
1= valeur compteur mémorisée dans le registre 1

IKResetREF
Cette fonction met le compteur à zéro à la marque de référence suivante.
Prototype: BOOL IKResetREF (USHORT Axis);
Axis: Numéro de l'axe (0 à 15)

IKStartREF
Cette fonction lance le compteur à la marque de référence suiv ante.
Prototype: BOOL IKStartREF (USHORT Axis);
Axis: Numéro de l'axe (0 à 15)

IKStopREF
Cette fonction stoppe le compteur à la marque de référence sui vante.
Prototype: BOOL IKStopREF (USHORT Axis);
Axis: Numéro de l'axe (0 à 15)

IKLatchREF
Cette fonction mémorise la valeur du compteur lors du franchis sement de la marque de référence.
Prototype: BOOL IKLatchREF (USHORT Axis, USHORT Latch);
Axis: Numéro de l'axe (0 à 15)

IKLatched

Cette fonction constate si la valeur du compteur a été mémoris ée. Application: Avant que la valeur du compteur soit lue, il faut demander si la valeur du compteur est mémorisée.

Prototype: BOOL IKLatched (USHORT Axis, USHORT Latch, BOOL* pStatus);

Axis: Numéro de l'axe (0 à 15)
Latch: 0 = interrogation pour registre 0
1 = interrogation pour registre 1
pStatus: Pointeur sur une variable Bool (16 bits)
false (= 0) = valeur non mémorisée
true (<> 0) = valeur mémorisée

IKWaitLatch

Cette fonction attend que la valeur du compteur ait été mémorisée. Application: Avant que la valeur du compteur soit lue, i l faut demander si la valeur du compteur est mémorisée.

Prototype: BOOL IKWaitLatch (USHORT Axis, USHORT Latch);

Axis: Numéro de l'axe (0 à 15)
Latch: 0 = interrogation pour registre 0
1 = interrogation pour registre 1

IKStrtCodRef

Cette fonction initialise le franchissement du point de référence (marques de référence à distances codées). Ensuite, interrogati on périodique (fonction: IKCodRef) **ou** attendre (fonction: IKWai tCodRef) que le franchissement des marques de référence à di stances codées soit achevé.

Prototype: BOOL IKStrtCodRef (USHORT Axis, USHORT Latch, ULONG RefDist);

Axis: Numéro de l'axe (0 à 15)
Latch: 0 = avec registre 0
1 = avec registre 1
RefDist: Ecart fixe entre les marques de référence
(ex. 500, 1 000, 2 000, 5000)

IKCodRef

Cette fonction constate si la 2ème marque de référence a été fr anchie (franchissement des marques de référence à distances codées) et délivre en retour la valeur d'offset. Celle-ci do it être additionnée à la valeur du compteur pour obtenir la positi on absolue. Il convient d'appeler cette fonction périodiquement après le lancement du franchissement du point de r éférence. Mais on peut aussi attendre la fin – fonction: IKWaitC odRef).

Prototype:
BOOL IKCodRef (USHORT Axis, BOOL* pStatus, double* pData);

Axis: Numéro de l'axe (0 à 15)

pStatus: Pointeur sur une variable Bool (16 bits).
False (= 0) = Franchissement non terminé.
True (<> 0) = Franchissement terminé.

pData:
Pointeur sur une „double variable“ (64 bits) dans laquelle se trouve la valeur d'offset (seulement si l'on a pStatus=TRUE).

IKWaitCodRef

Cette fonction attend jusqu'à ce que le franchissement du point de référence (marques de référence à distances codées) soit achevé. Après le franchissement de la deuxième marque de référence, la valeur d'offset est délivrée en retour.

Prototype: BOOL IKWaitCodRef (USHORT Axis, double* pData);

Axis: Numéro de l'axe (0 à 15)

pData:
Pointeur sur une „double variable“ (64 bits) dans laquelle se trouve la valeur d'offset.

IKStopCodRef

Cette fonction interrompt le processus de franchissement des marques de référence à distances codées.

Prototype: BOOL IKStopCodRef (USHORT Axis);

Axis: Numéro de l'axe (0 à 15)

IKClear

Cette fonction efface l'état d'erreur.

Prototype: BOOL IKClear (USHORT Axis);

Axis: Numéro de l'axe (0 à 15)

IKStatus

Cette fonction restitue l'état de l'IK□121.

Prototype: BOOL IKStatus (USHORT Axis, ULONG* pStatus);

Axis: Numéro de l'axe (0 à 15)

pStatus: Pointeur sur un mot long (32 bits)

Bit	Fonction
D0	1 = valeur compteur mémorisée dans le registre 0
D1	1 = valeur compteur mémorisée dans le registre 1
D2	sans fonction
D3	
D4	1 = compteur arrêté
D5	sans fonction
D6	1 = dépassement de fréquence
D7	sans fonction
D8	1 = fonction REF active
D9	1 = compteur lancé à marque de référence suivante[1)]
D10	1 = compteur stoppé à marque de référence suivante[1)]
D11	1 = compteur mis à zéro à la marque de référence suivante[1)]
D12	1 = compteur mémorisé à la marque de référence suivante[1)]
D13	1 = compteur mémorisé à la marque de référence après la suivante[1)]
D14	1 = compteur mis à zéro à chaque marque de référence [1)]
D15	sans fonction
D16	
D17 D18	Amplitude actuelle 00 = amplitude normale (4,5µA < U_e < 15µA) 01 = amplitude faible (U_e < 4,5µA) 10 = grande amplitude (U_e > 15µA) 11 = amplitude faible erronée (U_e < 2,5µA)

Bit	Fonction
D19 D20	Amplitude minimale 00 = amplitude normale (4,5µA < U_e < 15µA) 01 = faible amplitude (U_e < 4,5µA) 10 = grande amplitude (U_e > 15µA) 11 = amplitude faible erronée (U_e < 2,5µA)
D21 D22	Amplitude maximale **1)** 00 = amplitude normale (4,5µA < U_e < 15µA) 01 = faible amplitude (U_e < 4,5µA) 10 = grande amplitude (U_e > 15µA) 11 = amplitude faible erronée (U_e < 2,5µA
D23	sans fonction
D24 à D31	Code IC du circuit du compteur (08 ou 09)

1) seulement avec code IC = 09

IKRead32

Cette fonction délivre la valeur de compteur 32 bits. Avant que l a valeur du compteur puisse être lue, elle doit être mémorisée dans le registre 0 ou le registre 1 (IKLatch, IKLatchREF) et il faut interroger pour savoir si la mémorisation a eu lieu (IKLatched, I KWaitLatch).

Prototype: BOOL IKRead32 (USHORT Axis, USHORT Latch, LONG* pData);

Axis: Numéro de l'axe (0 à 15)
Latch: 0 = lecture à partir du registre 0
1 = lecture à partir du registre 1
pData: Pointeur sur un mot long (32 bits) dans lequel doit être classée la valeur du compteur.

IKRead48

Cette fonction délivre la valeur de compteur 48 bits. Avant que l a valeur du compteur puisse être lue, elle doit être mémorisée dans le registre 0 ou le registre 1 (IKLatch, IKLatchREF) et il faut interroger pour savoir si la mémorisation a eu lieu (IKLatched, I KWaitLatch).

Prototype: BOOL IKRead48 (USHORT Axis, USHORT Latch, double* pData);

Axis: Numéro de l'axe (0 à 15)
Latch: 0 = lecture à partir du registre 0
1 = lecture à partir du registre 1
pData: Pointeur sur une „double-Variable“ (64 bits) dans laquelle doit être classée la valeur du compteur.

IKReadPhase
Cette fonction lit le réglage actuel du potentiomètre de correcti on de phase.

Prototype: **BOOL IKReadPhase (USHORT Axis, BYTE* pData);**

Axis: Numéro de l'axe (0 à 15)

pData: Pointeur sur une „variable byte" (8 bits) d ans laquelle sera classée la correction de phase.

IKWritePhase
Cette fonction modifie le réglage actuel du potentiomètre de co rrection de phase.

Prototype: **BOOL IKWritePhase (USHORT Axis, BYTE Data);**

Axis: Numéro de l'axe (0 à 15)

Data: Nouvelle valeur de correction de phase (0 à 63)

IKLoadPhase
Cette fonction délivre en retour la valeur de correction de phase non volatile mémorisée.

Prototype: **BOOL IKLoadPhase (USHORT Axis, BYTE* pData);**

Axis: Numéro de l'axe (0 à 15)

pData: Pointeur sur une „variable byte" (8 bits) dans laquelle sera classée la correction de phase.

IKReadAmp
Cette fonction délivre en retour le réglage momentané de la cor rection d'amplitude.

Prototype: **BOOL IKReadAmp (USHORT Axis, BYTE* pData);**

Axis: Numéro de l'axe (0 à 15)

pData: Pointeur sur une „variable byte" (8 bits) d ans laquelle sera classée la correction d'amplitude.

IKWriteAmp
Cette fonction modifie le réglage momentané de la correction d 'amplitude.

Prototype: **BOOL IKWriteAmp (USHORT Axis, BYTE Data);**

Axis: Numéro de l'axe (0 à 15)

Data: Nouvelle valeur de correction d'amplitude (0 à 63)

IKLoadAmp
Cette fonction délivre en retour la valeur non volatile de correcti on d'amplitude mémorisée.
Prototype: BOOL IKLoadAmp (USHORT Axis, BYTE* pData);
Axis: Numéro de l'axe (0 à 15)
pData:
Pointeur sur une „variable byte“ (8 bits) d ans laquelle sera classée la correction d'amplitude.

IKReadOffset
Cette fonction délivre en retour le réglage momentané de la cor rection d'offset.
Prototype: BOOL IKReadOffset (USHORT Axis, SHORT* Ofs0, SHORT* Ofs90);
Axis: Numéro de l'axe (0 à 15)
Ofs0:
Pointeur sur une „short variable“ (16 bits) dans laquelle sera classée la correction d'offset d u signal 0 degré.
Ofs90:
Pointeur sur une „short variable“ (16 bits) dans laquelle sera classée la correction d'offset d u signal 90 degrés.

IKWriteOffset
Cette fonction modifie le réglage momentané de la correction d 'offset.
Prototype: BOOL IKWriteOffset (USHORT Axis, SHORT Ofs0, SHORT Ofs90);
Axis: Numéro de l'axe (0 à 15)
Ofs0:
Nouvelle valeur de correction d'offset du signal 0 degré (–63 à +63)
Ofs90:
Nouvelle valeur de correction d'offset du signal 90 degrés (–63 à +63)

IKLoadOffset
Cette fonction délivre en retour la valeur non volatible de correc tion d'offset mémorisée.
Prototype: BOOL IKLoadOffset (USHORT Axis, SHORT* Ofs0, SHORT* Ofs90);
Axis: Numéro de l'axe (0 à 15))
Ofs0:
Pointeur sur une „short variable“ (16 bits) dans laquelle sera classée la correction d'offset d u signal 0 degré.

Ofs90:

Pointeur sur une „short variable“ (16 bits) dans laquelle sera classée la correction d'offset du signal 90□degrés.

IKStore
Cette fonction transfère toutes les valeurs de correction actuell ement configurées vers une mémoire non volatile.
Les valeurs de correction de phases et d'amplitude sont activée s automatiquement par l'IK□121 lors de la mise sous tension du PC. Les valeurs de correction d'offset sont activées lors de l'ini tialisation de l'IK□121 (fonction: IKInit).
Prototype: BOOL IKStore (USHORT Axis);
Axis: Numéro de l'axe (0 à 15)

IKDefault
Cette fonction met toutes les valeurs de correction à des valeur s neutres (phase=31, amplitude=31 et offset=0). Ce état est pri s en compte dans la mémoire non volatile.
Prototype: BOOL IKDefault (USHORT Axis);
Axis: Numéro de l'axe (0 à 15)

IKRomRead
Cette fonction lit une valeur 8 bits à partir de l'EEPROM.
Prototype:
BOOL IKRomRead (USHORT Card, BYT E Adr, BYTE*pData);
Card: Numéro de l'IK□121 (0 à 7)
Adr: Adresse dans l'EEPROM (0 à 255)
pData:
Pointeur sur une „variable byte" (8 bits) d ans laquelle sera classée la valeur.

IKRomWrite
Cette fonction écrit une valeur 8 bits dans l'EEPROM.
Prototype:
BOOL IKRomWrite (USHORT Card, BY TE Adr, BYTE Data);
Card: Numéro de l'IK□121 (0 à 7)
Adr: Adresse dans l'EEPROM (0 à 255)
Data:
Valeur (8 bits) qui sera mémorisée dans l' EEPROM.

IKInputW
Cette fonction lit un mot d'un registre.
Prototype: BOOL IKInputW (USHORT Axis, USHORT Adr, USHORT* pBuffer);
Axis: Numéro de l'axe (0 à 15)
Adr: Adresse du registre (0 à 30 ou 0 à 0x1E)
pBuffer: Pointeur sur un mot (16 bits) dans lequel sera classée la valeur lue.

IKInputL

Cette fonction lit un mot long d'un registre.

Prototype: **BOOL IKInputL (USHORT Axis, USHORT Adr, ULONG* pBuffer);**

Axis: Numéro de l'axe (0 à 15)

Adr: Adresse du registre (0 à 28 ou 0 à 0x1C)

pBuffer: Pointeur sur un mot long (32 bits) dans lequel sera classé la valeur lue.

IKOutput

Cette fonction écrit un mot dans un registre.

Prototype: **BOOL IKOutput (USHORT Axis, USHORT Adr,**

Axis: Numéro de l'axe (0 à 15)

Adr: Adresse du registre (0 à 30 ou 0 à 0x1E)

Data: Valeur (16 bits) qui sera écrite dans le registre.

IKSetI2C

Cette fonction active les lignes du bus I^2C; par conséquent, vous pouvez activer ou désactiver les lignes de données et d'horloge. Ceci permet de contacter directement les potentiomètres et l'EEPROM.

Prototype: **BOOL IKSetI2C (USHORT Card, BOOL SCL, BOOL SDA);**

Card: Numéro de l'IK□121 (0 à 7)

SCL: Etat de la ligne d'horloge
FALSE(=0)= Low
TRUE(<>0)=High

SDA: Etat de la ligne de données
FALSE(=0)= Low
TRUE(<>0)=High

Caractéristiques techniques

Caractéristiques mécaniques

Dimensions	158 mm • 107 mm
Température:	
de travail	0°C à 45°C
de stockage	–30°C à 70°C

Caractéristiques électriques

Entrées/sorties

Signaux d'appel externes: X3: embase 4 plots

2 entrées:	U_H: 3,15 V à 30 V U_L: –3,0 V à 0,9 V
1 sortie:	U_H: 4,0 V à 32 V U_L: 0 V à 1,0 V
Sorties des systèmes: de mesure	signaux de courant sinusoïdaux (11 μA_{CC}) via kit supplémentaire pour chaque axe

IK 121 A

Entrées syst. mesure:	X1: axe 1, raccordement Sub-D 9 pl.; Signaux sinus: 7 μA_{CC} à 16 μA_{CC} X2: axe 2, raccordement Sub-D 9 plots; Signaux sinus: 7 μA_{CC} à 16 μA_{CC}
Fréquence d'entrée:	100 kHz max.
Longueur de câble:	10 m max.

IK 121 V

Entrées syst. mesure:	X1: axe 1, raccordement Sub-D 15 pl. Signaux sinus: 0,6 V_{CC} à 1,2 V_{CC} X2: axe 2, raccordement Sub-D 15 plots; Signaux sinus: 0,6 V_{CC} à 1,2 V_{CC}
Fréquence d'entrée:	400 kHz max.
Longueur de câble:	30 m max.

Câbles possibles jusqu'à 150 m si l'alimentation externe garantit 5 V sur le système de mesure. Dans ce cas, la fréquence en entrée est réduite à 250 kHz.

Interpolation du signal par 1024

Alignement des signaux des systèmes de mesure	**Phase et amplitude à l'aide de potentiomètres électroniques** **Offset à l'aide des registres situés dans les compteurs**
Registres de données pour valeurs mesure	48 bits
Adresses de port	configurables au moyen de commutateurs DIP; l'IK 121 possède 16 adresses
Interruptions	IRQ5, IRQ9, IRQ10, IRQ11, IRQ12 ou IRQ15
Consommation	env. 1 W, sans système de mesure

Logiciel	
Logiciel driver et programmes de démonstration	comme aide à la programmation pour: **applications DOS** en „TURBO PASCAL“ et „BORLAND C++“ **WINDOWS NT / 95** en VISUAL C++, VISUAL BASIC et BORLAND DELPHI

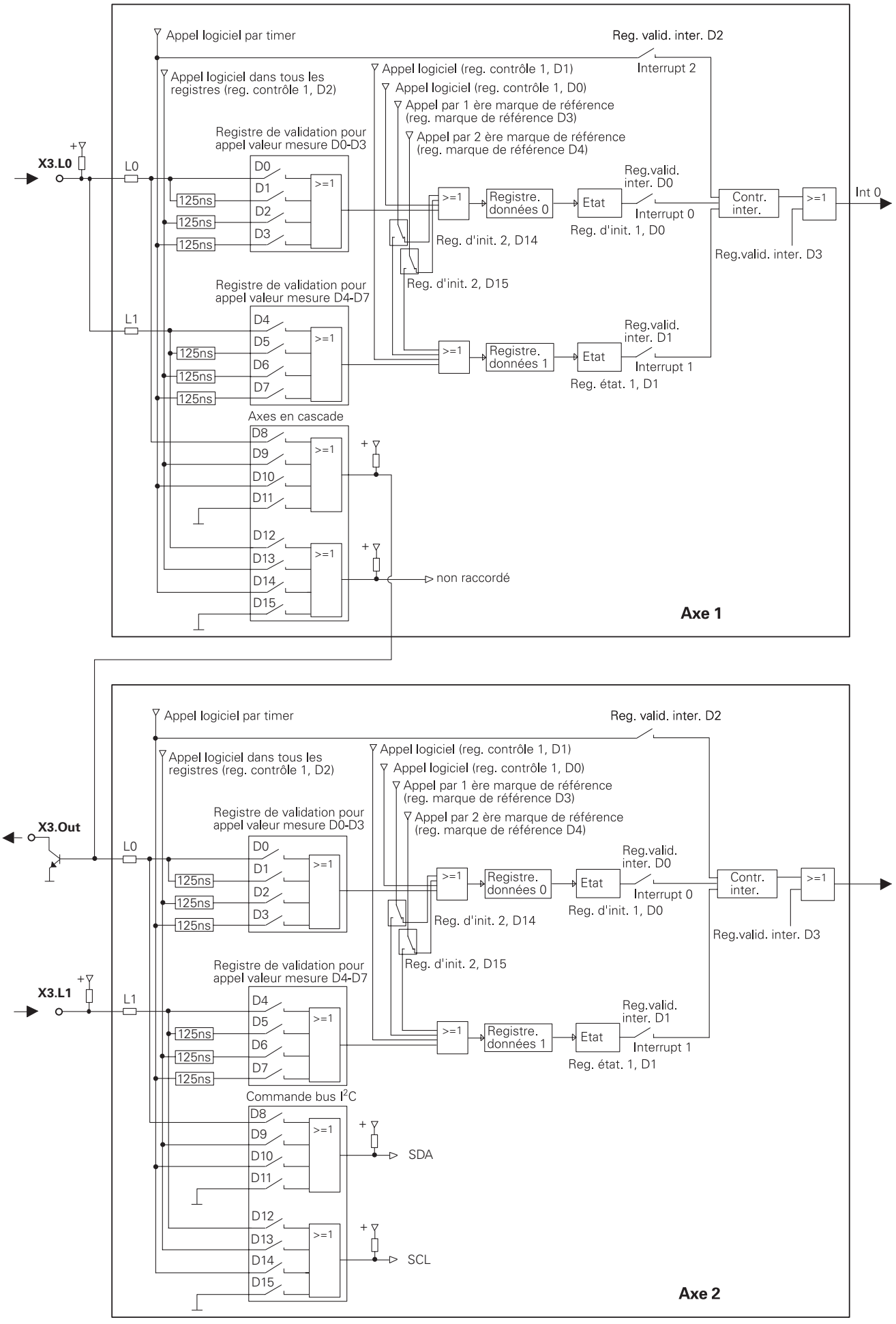
Index

Synoptique modulaire des voies d'appel dans les compteurs

Le synoptique suivant illustre:

- l'effet des signaux d'appel sur les registres de données
  la fonction des différents bits du registre de validati
- on pour l'appel de la valeur de mesure
  les axes en cascade avec les bits correspondants d
  u registre de même nom
- la formation des interruptions
- le registre de commande du bus I²C

Les circuits de retard (retard de 125 ns) sont utilisés pour la mé morisation synchrone des deux axes afin d'équilibrer la durée de transit du signal entre l'axe 1 et l'axe 2. Lors de la mémorisatio n synchrone, il est souhaitable, pour l'axe 1, de sélectionner une voie de signal par circuit de retard et, pour l'axe 2, sans circuit d e retard. Dans la mesure où l'axe 1 et l'axe 2 utilisent le même compteur, des circuits de retard sont présents sur les deux axes ; cela veut dire que toutes les combinaisons de voies du signal n e donnent pas un appel significatif!


Appel logiciel par timer
Reg. valid. inter. D2
Appel logiciel dans tous les registres (reg. contrôle 1, D2)
Appel logiciel (reg. contrôle 1, D1)
Interrupt 2
Appel logiciel (reg. contrôle 1, D0)
Appel par 1 ère marque de référence (reg. marque de référence D3)
Appel par 2 ère marque de référence (reg. marque de référence D4)
Registre de validation pour appel valeur mesure D0-D3
X3.L0
L0
D0
D1
D2
D3
125ns
>=1
Registre. données 0
Etat
Reg.valid. inter. D0
Interrupt 0
Reg. d'init. 1, D0
Contr. inter.
Int 0
Reg. d'init. 2, D14
Reg.valid. inter. D3
Reg. d'init. 2, D15
Registre de validation pour appel valeur mesure D4-D7
L1
D4
D5
D6
D7
Registre. données 1
Reg.valid. inter. D1
Interrupt 1
Reg. état. 1, D1
Axes en cascade
D8
D9
D10
D11
D12
D13
D14
D15
non raccordé
Axe 1
X3.Out
X3.L1
Commande bus I²C
SDA
SCL
Axe 2

HEIDENHAIN

DR. JOHANNES HEIDENHAIN GmbH
Dr.-Johannes-Heidenhain-Straße 5
83301 Traunreut, Germany
☏ +49 (8669) 31-0
FAX +49 (8669) 5061
e-mail: info@heidenhain.de

Technical support FAX +49 (8669) 31-1000
e-mail: service@heidenhain.de
Measuring systems ☏ +49 (8669) 31-3104
e-mail: service.ms-support@heidenhain.de
TNC support ☏ +49 (8669) 31-3101
e-mail: service.nc-support@heidenhain.de
NC programming ☏ +49 (8669) 31-3103
e-mail: service.nc-pgm@heidenhain.de
PLC programming ☏ +49 (8669) 31-3102
e-mail: service.plc@heidenhain.de
Lathe controls ☏ +49 (711) 952803-0
e-mail: service.hsf@heidenhain.de

www.heidenhain.de

291 019-33· 5 · 10/98· F&W · Printed in Germany · Sous réserve de modifications