HAMEG®
Instruments

HO80

IEEE-488 Interface Karte
für
IBM®PC - XT - AT
und kompatible Computer

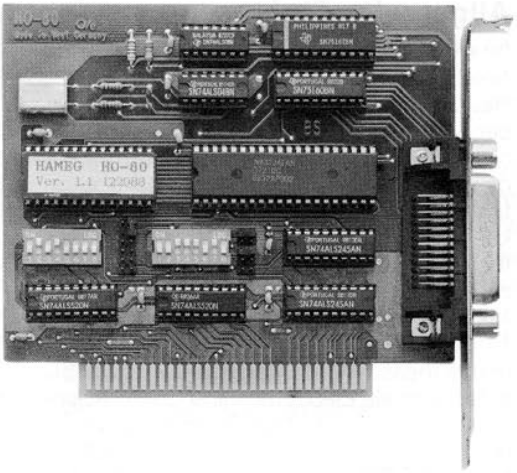
Inhaltsverzeichnis

Printed in Germany

Allgemeiner Hinweis:

Änderungen in Hard- oder Software, welche nach Drucklegung dieses Handbuches erfolgten, entnehmen Sie bitte den Liesmich oder Readme Files auf der Diskette.

HAMEG HO80



Technische Daten:

Bus:	PC/XT/AT 1/2 Slot Karte
Anschluß:	24-polige Buchse
Controller:	NEC µPD7210C
Systemadresse:	H2B
ROM-Adresse:	H4000 - HE000
DMA Kanal:	1 oder 3 wählbar
Interrupt:	zwischen 2 und 7 einstellbar
Hochsprachen:	BASICA, GWBASIC, MS-QuickBASIC und MS-BASIC Compiler Turbo-Pascal -4.0, -5.0, -5.5, -6.0, Turbo-C, MS-C und MS-Quick-C
Lieferumfang:	Interface-Karte, Software und Handbuch in Deutsch.

IEEE-488 Interfacekarte

Die HAMEG HO80 Interfacekarte ist für **IBM-PC/XT/AT** und 100% kompatible Rechner konzipiert. Der Bus-Controller NECµPD7210C ist das Hardware-Kernstück des HO80 Interface. Es unterstützt alle notwendigen Interrupt und DMA-Möglichkeiten des PC's.

Die Treibersoftware unterstützt **BASICA, GWBASIC, MS-Quick-BASIC, Borland Turbo-Pascal, MS-QuickC und MS-C.** Die Treibersoftware ist für das PC-Betriebssystem MS-DOS(PC-DOS) ab Version 2.1 ausgelegt.

Sämtliche Interrupt- und DMA-Möglichkeiten sind, unter Einbindung der im Handbuch beschriebenen Routinen bzw. auf der Diskette befindlichen Programme, nutzbar.
Die HAMEG HO80 Interface-Karte ist in der Lage, schnellen Daten-Blocktransfer (DMA) mit einer Geschwindigkeit - je nach Rechnertype und Leitungslänge - bis ca. 25 KByte/Sek (PC 4.7 MHz Takt) zu übertragen. Der Datenblock darf max. 64 KByte lang sein.

Auf der Installationsdiskette befinden sich Demo-Programme für den Byte- und Block-weisen Datentransfer, das Arbeiten mit SEC (Sekundär) Adressen, sowie die Möglichkeit der Interruptabfrage SRQ (Service Request). Diese Programme sind speziell für HAMEG-Geräte geschrieben. Die Interface-Karte HO80 unterstützt jedoch nicht nur HAMEG-Geräte, sondern alle IEEE-Busgeräte welche nach dem IEEE-488 / IEC 625 Standard arbeiten.

Für den Einsatz mit IEC 625 Geräten ist ein entsprechender Adapter zur Umsetzung der Stecker-Norm notwendig. Zur Verbindung zwischen der **HAMEG HO80** Interface-Karte und **HAMEG IEEE-488** Geräten wird ein IEEE-488-Bus-Kabel, mit der **HAMEG Zubehör Nr. HZ72**, benötigt.

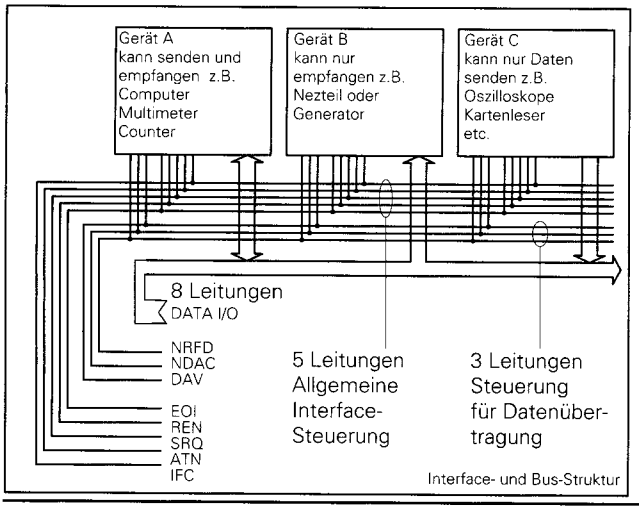
Allgemeine Einführung

Im Jahre 1975 drangen die ersten Informationen über den IEC-Bus an die Öffentlichkeit. Heute hat sich dieses System auf dem Gebiet der Meßtechnik weltweit durchgesetzt. Ein nicht unwesentlicher Anteil der heute neu auf den Markt kommenden Geräte, sind systemfähig mittels IEC-Bus. Jeder Meßtechniker kommt früher oder später in die Lage, sich mit IEC-Busgesteuerten Automaten auseinanderzusetzen.

Mit der Verabschiedung der Publikation 625-1 "Standard Interface System for programmable measuring equipment" durch die IEC (Internationale Electrotechnical Commission) steht solch ein Interface-System zur Verfügung. Für dieses Bus-System haben sich im Laufe der Zeit leider verschiedene Namen eingebürgert. Neben der international genormten Bezeichnung IEC-625-Bus (kurz: IEC-Bus) ist auch der amerikanische Name IEEE488/75 üblich. Bis auf den Stecker sind beide Normen identisch. Die Stecker der amerikanischen Norm sind 24polig, während die Internationale Norm einen 25poligen Stecker vorschreibt. Auf dem Markt sind jedoch Adapter dafür erhältlich. Entgegen der Empfehlung des Normenausschusses setzt sich jedoch der Stecker der amerikanischen Norm immer mehr durch. Die HAMEG HO 80 Schnittstelle besitzt ebenfalls den 24poligen Stecker nach amerikanischer Norm.

Die Bus-Elektronik ist jeweils im Gerät untergebracht. Der IEEE-Bus arbeitet mit TTL-Pegel in der "Negativ-True"-Logik (logisch 0 entspricht high Pegel). Die Ausgangstreiber haben ein "Fan-Out" von 30mA. Damit werden die "Thresh-Hold-Pegel" (Schwellwerte) auch bei voller Geräteanzahl erreicht. Das Impedanzverhalten der Verbindungskabel (spezifiziert in der Norm) unter Berücksichtigung der Signalzeiten läßt ein maximale Bus-Länge von 20m zu, mit der Einschränkung, daß pro Gerät eine Kabellänge von 2m nicht überschritten werden soll. Das bedeutet, daß alle 2m eine ohmsche Last (Wirklast) vorhanden sein muß. Die Übertragungsgeschwindigkeit von einem Mega-Byte/sec. wird erreicht bei einer Kabellänge pro Gerät von 0,5m, und wenn 48mA Tri-State-Treiber verwendet werden. Die tatsächliche Maximalgeschwindigkeit hängt natürlich sehr stark von dem Zeitverhalten der angeschlossenen Geräte ab. Die höchsten Datenraten werden nur dann erzielt, wenn in den Geräten entsprechende Zwischenspeicher vorhanden sind. An eine Schnittstelle HO 80 können bis zu 15 Geräte gleichzeitig angeschlossen werden. Die Anschlußfolge kann linear oder sternförmig erfolgen.

Als Anhaltspunkt für die Übertragungsgeschwindigkeit von der Controllerseite kann angenommen werden, daß beim Einsatz eines PC-XT Rechners mit einer Übertragungsrate von ca. 5 bis 25 K-Byte/sec. zu rechnen ist.



Interface- und Bus-Struktur

Struktur des IEEE-Interface-Systems

Die Verbindung der einzelnen Geräte wird von einem Bus-Kabel mit 16 parallelen. Leitungen vorgenommen. Über diese Leitungen werden Daten ausgetauscht, Steueranweisungen übermittelt und der Datenfluß überwacht (Handshaking). Die angeschlossenen Geräte werden nach ihren Funktionen in Controller, Talker und Listner eingeteilt.

Talker, Listener und Controller

Ein Gerät mit Talker-Funktionen kann Daten an andere Teilnehmer aussenden, z.B. ein Voltmeter oder Zähler. Geräte, die nur Daten empfangen können werden als Listener bezeichnet (z.B. Stromversorgungen und Signalgeneratoren). Es gibt auch Geräte, die beide Funktionen beherrschen. Der Controller, ein Rechner, überwacht und steuert den Datenaustausch zwischen den Geräten. Das notwendige Computer-Interface ist die Hameg Schnittstelle HO 80 für IBM PC/AT und kompatible MS DOS Computer.

Datenbus

Acht der 16 Bus-Leitungen bilden den Datenbus und werden zum Senden von Daten, Adressen, Programmbefehlen, Statusbyte und speziellen Buskomandos verwendet. Auf dem Datenbus unterscheidet man zwei Übertragungsarten: den Datenmode und den Befehlsmode. Die Kontrolleitung ATN signalisiert mit ihrem Zustand Low= Datenmode High= Adreßmode. Adressen und Buskommandos werden immer vom Controller ausgesendet.

Kontroll-Bus

Jede der fünf Leitungen des Kontroll-Busses hat eine besondere Funktion:

ATN = attention (Controller Befehl zur Deklaration)
a) einer Adresse
b) "unlisten"-Befehle
c) "polling"-Befehle

IFC = interface clear (Controller Befehl an alle Geräte)
Setzen aller angeschlossenen Geräte in einen definierten Zustand.

REN = remote enable
Dient der Überwachung und Fernsteuerung von (remote control) der angeschlossenen Instrumente.

EOI = end or Identify
Kann als letztes Zeichen einer Gesamtinformation gesendet werden.
SRQ = service request
Wird von irgend einem Gerät aktiviert. Wenn diese Leitung High ist, weiß der Controller, daß ein Gerät ihm etwas mitteilen möchte (seriell-polling). Der Controller kann dann das laufende Programm unterbrechen und die Nachricht des Gerätes abholen.

Der Handshake-Bus

Die Übertragung von Informationen auf dem Datenbus wird durch die Aktivitäten des Handshake-Bus begleitet. Es gibt drei Handshake-Leitungen:

DAV = data valid (Daten sind gültig)
Mit "low" wird vom "talker" signalisiert, daß die auf den Datenleitungen befindlichen Daten gültig sind

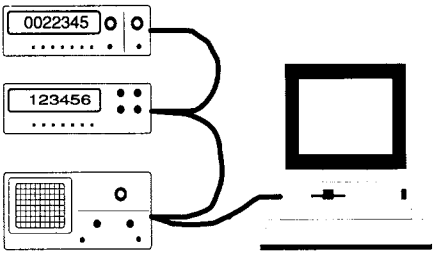
NRFD = not ready for Data (nicht empfangsbereit)
Die "listner" geben über diese Leitung ihre Empfangsbereitschaft bekannt. Es gilt: High= ready (Empfangsbereit), Low= **N**ot **R**eady **F**or **D**ata (nicht Empfangsbereit).

NDAC = not data accepted (Information nicht aufgenommen)
Mit "high" quitiert ein "listener" die Übernahme der anliegenden Daten.

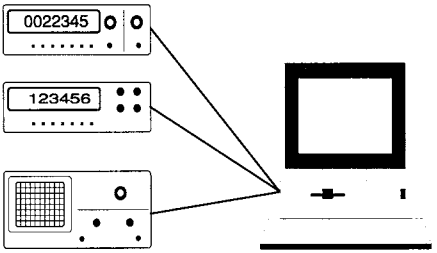
Geräte Adresse und Verbindungart

Alle an den IEEE-Bus angeschlossenen Geräte müssen unterschiedliche Geräteadressen aufweisen. In den Demoprogrammen verwenden wir für Oszilloskope die Adresse 6. Dem Computer (IEEE-Controller) wurde die Adresse 21 zugeordnet.

Verbindungsart Linear



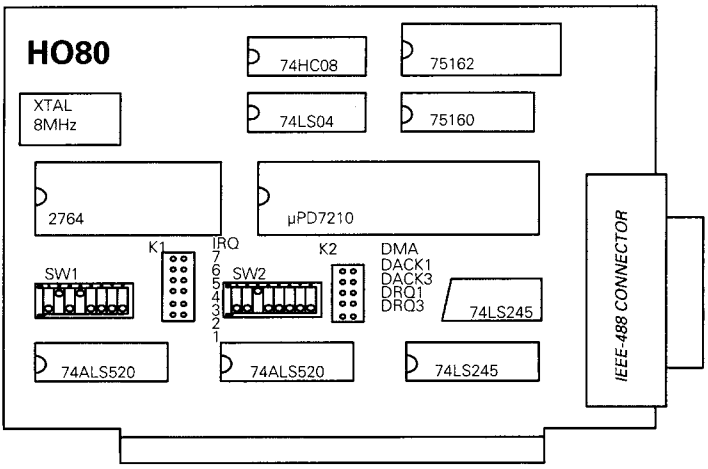
Verbindungsart Stern



Installation der HAMEG IEEE Schnittstelle HO 80

Die Hameg Schnittstelle HO 80 kann in alle PC oder AT Slots mit Ausnahme des Slots Nr. 8 bei PC-XT eingesetzt werden.

Der Einbau darf nur bei ausgeschaltetem Computer erfolgen.
Vor dem Einsetzen sind die Jumpers und Schalter einzustellen.
Die werkseitige Einstellung bei Lieferung ist für den SW 2: Speicheradresse &HDE00
SW 2: Pos 8 off = System Controller on für den SW 1: I/O Adresse &H2B8
K 1 Interrupt nicht gesetzt, K 2 DMA Kanal 3 ist aktiv. (PC-AT mit Harddisk!)



Änderung der Schnittstellenparameter

Die ROM-Adresse ist entsprechend dem Gesamtspeicher so zu wählen, daß die Adresse über dem vorhandenen RAM zuzüglich dem der Grafikkarte liegt. Dabei ist zu beachten, daß das BIOS-ROM bei PC ab Speicherbereich &HC700 bis &HC800 und bei AT Computern im Bereich ab &HE000 liegt. Mit den Dipschaltern am SW 2 ist die gewünschte Adresse einzustellen. Die Adresse kann in 8K Byteschritten eingestellt werden. Die Schalterpositionen haben folgende Wertigkeiten.

0 = Schalterstellung ON / 1 = Schalterstellung OFF

Position	1	2	3	4	5	6	7
Adresse	A19	A18	A17	A16	A15	A14	A13
Wert	512K	256K	128K	64K	32K	16K	8K

somit errechnet sich die eingestellte Adresse

BIN 1 1 0 0 0 0 0 = &HC000
768K = 512 + 256
BIN 1 1 0 0 0 1 0 = &HC400
784K = 512 + 256 + 16
BIN 1 1 0 1 0 0 0 = &HD000
832K = 512 + 256 + 64
BIN 1 1 0 1 1 1 1 = &HDE00
888K = 512 + 256 + 64 + 32 + 16 + 8

Die Wahl der Bereiche HA00 bis HB00 und der Adresse HE00 ist nicht empfehlenswert.

Einstellen der I/O Adresse mit dem Schalter SW 1.

Dieser Schalter setzt den Decoder der I/O Leitungen des NEC µPD7210C Controllers. Für viele Anwendungen ist eine Änderung der Adresse nicht notwendig. Die Schalter vom SW 1 können die Adresse an den Leitungen A9 bis A3 verändern. Ist der Schalter geschlossen (on) so ist die Signalleitung = 0. Ist der Schalter offen (off) so ist die Signalleitung = 1

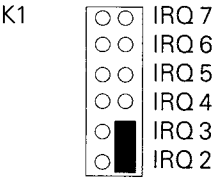
SW 1 Position 7 A3 off = 1
SW 1 Position 6 A4 off = 1
SW 1 Position 5 A5 off = 1
SW 1 Position 4 A6 on = 0
SW 1 Position 3 A7 off = 1
SW 1 Position 2 A8 on = 0
SW 1 Position 1 A9 off = 1

Leitung	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Wert	1	0	1	0	1	1	1	0	0	0
HEX	2		B		8					

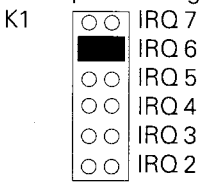
Einstellen des Interruptes

Bei Lieferung ist kein Interrupt eingeschaltet.
Die Einstellung des Interrupts geschieht mittels Kurzschlußstecker auf der Stiftreihe K 1
Zwei Beispiele zeigen die Möglichkeiten auf.

Kein Interrupt eingestellt



Interrupt Nr. 6 eingeschaltet

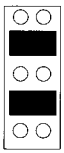


Einstellen des DMA Kanals

Als DMA Kanäle können beim NEC µPD7210C der DMA Kanal 1 und der DMA Kanal 3 ausgewählt werden. Für PC-XT empfehlen wir DACK1 und DRQ1 zu setzen. Die Einstellung für PC-AT sollte DACK3 und DRQ3 sein.
Ist eine Änderung des DMA Kanals erforderlich, so setzen Sie die Kurzschlußbrücken am Stiftblock K 2.

Ansicht: DMA Kanal 3 eingeschaltet.

K2



DACK 1
DACK 3
DRQ 1
DRQ 3
N.C. nicht benutzt

Einstellen des System Controllers

Die Funktion System Controller ein / aus wird an dem Schalter SW 1 Position 8 eingestellt.

SW 2 Pos. 8 off: Rechner ist System Controller.

SW 2 Pos. 8 on: Rechner ist kein System Controller.
Ein zweiter Rechner oder ein Gerät übernimmt die Funktion als System Controller. Beim Einsatz von einem Rechner sollte dieser stets als System Controller konfiguriert sein.

Inhalt der Softwarediskette

Basic Konfigurations und Demoprogramme
Konfigurations Programme LOAD8000-LOADDE00 in Sub-DIR BASICDRV
Basic-Programme in Sub-DIR BASIC

Initialisierung, Senden und Empfangen von Daten	ISE.BAS
Service Request und Seriell Poll	SRQ.BAS
Sekundäradresse und Datenblock lesen	IRDA.BAS
Datenblock schreiben	IWDA.BAS

Turbo-Pascal Programme in Sub-DIR PASCAL
HO80_1.DOC, HO80_1.P40, HO80_1.P50, HO80_1.P55 und READ.ME
IEEEPAS1.PAS
IEEEPAS2.PAS
C-Programme in Sub-DIR C
CL-, CM- und CSCOMMON.ASM
CL-, CM- und CSCOMMON.OBJ
DEMO208.C und DEMO208.EXE, HM8112.C und HM8112.EXE für MS-C
HM8122.C und HM8112.EXE für Turbo-C
Bezeichnerliste MC.H für MS-C und Turbo-C
Object-Codes in Sub-DIR C\MASM für Microsoft MACRO-ASSEMBLER
Object-Codes in Sub-DIR C\TASM für Borland Turbo-Assembler

Befehlsliste HO 80

Basic Befehl:	Offset der Adressen Interpreter	/ Compiler
Call **INIT** (adress%,pegel%) Initialisierung adress% = Adresse vom PC, z.B. 21 pegel% = 0 System Controller 1 Aktiver Controller 2 Gerät	0	0
Call **IWSD** (komando$,status%) Sende sekundär Adresse komando$ = Befehls- oder Datenwort Beispiel: "MLA TALK 6 SEC 15"	3	30
Call **IRSD** (e$,lang%,status%) lese sekundär adresse e$ = Datensatzeingang lang% = Datensatzlänge status% = gibt den Status für alle Routinen 0 = Datentransfer ohne Fehler 8 = Zeitüberlauf 32 = Transfer ende mit EOI	6	33
Call **IWD** (instrument%,befehl$,status%) Sende Datenwort instrument% = Geräte Adresse befehl$ = Befehlsstring	9	36
Call **IRD** (e$,lang%,instrument%,status%) lese Datenwort e$ = empfangener Datensatz lang% = Datensatzlänge instrument% = Geräte Adresse	21	39
Call **ISPL** (instrument%,pol%,status%) instrument% = Geräteadresse für Poll pol% = Status - Byte Poll	12	12
Call **IPPL** (pol%) pol% = Status - Byte Poll	15	15
Call **IWDA** (segment%,offset%,anzahl%,eoi%,status%) Sende Datenarray segment% = Segment Adresse der Daten offset% = Offsetadresse der Daten anzahl% = Anzahl der zu übertragenen Byts eoi% = 1 oder 0 für mit und ohne Endzeichen	200	200
Call **IRDA** (segment%,offset%,anzahl%,lang%,status%) lese Datenarray segment% = Segment Adresse der Daten offset% = Offsetadresse der Daten anzahl% = Array größe lang% = gelesene Datensatzlänge	203	203

Zusammenfassung der direkt Kommandos

Die direkt Kommandos des IEEE-Bus teilen sich in vier Gruppen auf. Sie unterscheiden sich in: Universalbefehle, Mehrdrahtnachrichten, adressierte Befehle und Sekundärbefehle.

Universalbefehle

IFC = Interface Clear	/ *Schnittstellenfunktion rücksetzen.*
REN = Remote Enable	/ *Fernsteuerung freigeben.*
ATN = Attention	/ *Achtung.*
EOI = End of Output/identify	/ *Erkennung des Datenende.*
END = EOI	/ *mit zusätzlichem CR CHR$(13).*

Mehrdrahtnachrichten

DCL = Device Clear	/ *Gerät rücksetzen.*
LLO = Local Lockout	/ *Steuerung verriegeln.*
PPU = Parallel Poll	/ *Parallelabfrage abschalten.*
SPE = Seriell poll enable	/ *Serienabfrage freigeben.*
SPD = Seriell poll disable	/ *Serienabfrage sperren.*
UNL = Unlisten	/ *Listener wird inaktiviert.*
UNT = Untalk	/ *Talker wird inaktiviert.*

Adressierte Befehle

GET = Group Execute Trigger	/ *Gerätegruppe auslösen.*
SDC = Selected Device Clear	/ *angewähltes Gerät rücksetzen.*
GTL = Goto Local	/ *Auf Eigensteuerung schalten.*
PPC = Parallel Poll Configure	/ *zur Parallelabfrage konfigurieren.*
TCT = Take Control	/ *Steuerung übernehmen.*

DATA = Senden einer Dateninformation.
LISTEN = Definiert ein Listener (*Hörer*) oder Listener mit Adresse.
TALK = Definiert ein Talker (*Sprecher*) mit Adresse.

Sekundärbefehle

PPE = Parallel Poll Enable	/ *Parallelabfrage freigeben*
PPD = Parallel Poll Disable	/ *Parallelabfrage sperren.*

SEC = Secundäradressen-Übergabe / *Als Erweiterung reiner Listener oder Talker Geräte wie z.B. HAMEG Oscilloscope HM 208.*

PC-Befehle

PC Befehle sind für die Einleitung besondere Betriebsarten erforderlich. Sie leiten den Datentransfer zwischen zwei PC's ein oder das Senden von SEC-Befehlen

MLA = MY.Listen Adresse. PC wird Listener
MTA = MY.Talk Adresse. PC wird Talker

Testprogramme

LOOKHO80.EXE
Schnittstellentestprogramm zur Initialisierungsprüfung:
Als Antwort erhalten Sie bei einwandfreier Funktion** Ver. 1.1 ** sowie die auf der Schnittstellenkarte eingestellte Adresse des Eproms. Wird die Karte nicht gefunden, so wird eine Fehlermeldung ausgegeben.

TESTHO80.EXE
Senden von einzelnen Befehlen und Empfang von Daten in Einzelschritten mit dem Programm:
Nach dem Programmstart können Sie unter 3 verschiedenen Landessprachen wählen. Durch Eingabe des dem Landeskenner zugeordneten Buchstabens, wird das Programm in der Landessprache gestartet. Zur Ermittlung der Kartenadresse läuft zuerst ein Selbsttest. Wird die HO80 Karte gefunden, wird dies auf dem Bildschirm mit zugehöriger Adresse angezeigt. Sollte auf der Rechnerseite ein Defekt vorliegen, wird eine Fehlermeldung ausgegeben. Defekte im Bereich des IEEE-Buses werden nicht erfaßt. Für die Funktion SRQ und seriell POLL ist die Adresse des NEC µPD7210C Controllers auf den Wert &H2B8 fest eingestellt. Der Computer belegt die Adresse 21.

Nach dem Selbsttest wird ein Input verlangt. Wird an dieser Stelle ein Fragezeichen '?' eingegeben, so werden die Datentransfer-Modi angezeigt.
Folgende Eingaben bewirken den Aufruf des Programmteils:

'S' für sende Befehl zum Gerät
'E' für empfange Daten vom Gerät
'T' für sende Direkt- und SEC- Kommandos
'O' lese Datenarray vom Oscilloscope
'R' für empfange Direkt- und SEC- Kommandos
'P' für SRQ und seriell Poll
'Q' für Ende

Die zu Übertragenen Datensatzlängen für Senden und Empfang sind auf 80 Zeichen eingestellt. Eine Ausnahme macht dabei das Programmteil 'O'. Es wurde speziell auf die Bedürfnisse der Hameg Oscilloscope HM 205-2 und HM 208 eingestellt. Hier kann ein Datenblock bis zu einer Gesamtlänge von 2048 Bytes Übertragen und als Zahlenwerte angezeigt werden. Für den Datentransfer mit SEC (Sekundäradressen) gilt folgendes Beispiel:

MLA = Computer sendet SEC-Kommando zum Einlesen
Eingabe SEC Kommando **MLA TALK 6 SEC 10**
TALK 6 = an Geräteadresse 6
SEC 10 = Sekundärkommando 10 (Oscilloscope-Daten als Daten Block auslesen).

MTA = Computer sendet SEC-Komando zum Aufsühren einer Funktion.
Eingabe SEC Kommando **MTA TALK 6 SEC 4**
SEC 4 = Sekundärkommando 4 (Oscilloscope RESET setzen).

Portumleitung

Anwendung der HO 80 Schnittstelle mit einem IEEE Drucker oder Plotter.
Möchten Sie einen Drucker oder Plotter mit IEEE-Bus an Ihrer Systemeinheit installieren, so ist dies möglich. Mit dem Programm PRN_IEEE.EXE, welches sich auf der Diskette befindet, können Sie den Printer Port LPT1: auf den IEEE-Bus umschalten. Verwenden Sie folgende Syntax:
Für ein Gerät mit Listener Adresse 5 starten Sie

PRN_IEEE 5

alle nun zum Port PRN: oder LPT1: gesendeten Daten werden von PRN: über das HO 80 IEEE Interface zur IEEE-Adresse 5 umgeleitet.

Basic-Interpreter

Konfigurierung der Schnittstelle bei BASIC-Programmierung mit den Programmen
LOADC400.COM
Initialisiert die Schnittstellenkarte mit der Adresse &HC400,
LOADC800.COM
Initialisiert die Schnittstellenkarte mit der Adresse &HC800.
Mit der Initialisierung wird im RAM-Bereich der notwendige Speicherplatz reserviert, um eine einwandfreie Funktion sicherzustellen. Dieses gilt gleichermaßen für die Adressen **H4000, H8000, HC000, HD000 und HDE00.**

Quick-Basic & Basic-Compiler

Die HO 80 IEEE Karte kann unter Quick-Basic und Basic-Compiler betrieben werden. Hierfür sind alle Anwendungsprogramme nur in 2 Punkten umzustellen.
1. Die Offsetadressen müssen von Interpreter- in Compiler-Werte geändert werden.
2. Der Befehl steht nicht mehr hinter dem CALL Aufruf sondern ist Teil des Kommandostrings.
zum Beispiel:
Interpreter Basic
10 IWD = 9
20CALL **IWD**(instrument%,befehl$,status%)

Compiler
IWD% = 36
CALL **ABSOLUTE**
(instrument%,befehl$,status%,**IWD%**)
Für den Einsatz mit dem Quick-Basic Compiler 2.0 befindet sich ein USERLIB.EXE Files auf der Diskette in der Sub-Directory QBASIC2. Dieses von uns erstellte USERLIB.EXE Programm beinhaltet den ABSOLUTE Befehl. für die QBASIC Versionen 2.0 und 2.01. Verwenden Sie Quick-Basic 4.0 oder 4.5 so gelten einige Einschränkungen. Der Befehl Call Absolute in Verbindung mit Sekundäradressen (IWSD%) wird nicht von der QB4.xx-Benutzeroberfläche ausgeführt. Erst nach der Generierung eines EXE-Files ist das Programm lauffähig.

Turbo-Pascal 4.0, 5.0, 5.5 & 6.0

Für die Anwendung der Hameg HO80 IEEE-Schnittstellenkarte befindet sich ein Assembler Programm mit dem Namen **HO80_1.TPU** auf der Softwarediskette. Dieses in Turbo Pascal einzubindende Programm stellt die automatische Erkennung und Initialisierung, sowie die gesamten Befehle der HO80 Schnittstelle zur Verfügung. Programmierhinweise finden Sie in den mitgelieferten Demoprogrammen IEEEPAS1 und IEEEPAS 2.

Erstellung eines TPU-Files

Für die Versionen Turbo-Pascal 4.0, 5.0 und 5.5 sind drei Programme auf der Softwarediskette mit den Namen HO80_1.P40, HO80_1.P50 und HO80_1.P55. Sie müssen durch COPY das Programmfile HO80_1.TPU erzeugen. Verwenden Sie hierzu den COPY Befehl:
z.B. für Turbo-Pascal 4.0
COPY HO80_1.P40 HO80_1.TPU
zur Erstellung des .TPU Files.

Softwaretools für MS-C und Turbo-C

Zur Unterstützung vom Microsoft C, Quick C Version 1.01B und Borland Turbo C sind auf der Softwarediskette im Unterverzeichnis C folgende Programme enthalten:

CSCOMMON.ASM, CSCOMMON.OBJ, MC.H. DEMO208.C, HM8112.C und HM8122.C als Source-Code, *.OBJ Datei und *.EXE File. In den Unterverzeichnissen MASM und TASM befinden sich die Assembler-Codes entsprechend MASM=Microsoft- oder TASM=Turbo-Assembler. Das Programm CSCOMMON.ASM im Sourcetext dient fortgeschrittenen C-Programmierern als Referenzmöglichkeit zur Lösung eigener Problemstellungen. Es enthält die Assembler - Routinen für Microsoft QuickC 1.01B, zur Unterstützung der HAMEG Interface-Karte HO80. Bei Verwendung der Bezeichnerliste, im Programm MC.H enthalten, genügt es das vorhandene CSCOMMON.OBJ File mit einzubinden. Das hier verwendete Speichermodell ist **SMALL**. Zum Kompilieren und Linken der Module können Sie folgende Befehlszeile eingeben :

```
; qcl /AS /F 1000 DEMO208.C CSCOMMON.OBJ
```

Gleiches gilt für die Speichermodelle Medium und Large. Sie werden von den Modulen "CMCOMMON.OBJ" bzw. "CLCOMMON.OBJ" unterstützt. Die Option für "QCL" ist /AM für "Medium" und /AL für "Large". Das Programm DEMO208.C ist ein Demonstrationsprogramm für die Steuerbefehle des HAMEG Oscilloscope HM 208 und HM205-2. Zur Einbindung ist dieses Programm gleichzeitig auch als .OBJ File auf der Diskette vorhanden.

Programmbeschreibungen

INITIALISIERUNG

In jedem BASIC-Programm muß stets ein besonderer Syntax eingehalten werden. Dieser besteht aus der Speicheradresse, dem Unterroutinenoffset und der Angabe über die Controllerpriorität. Die Speichersegmentadresse ist der Wert, welcher mit den DIP-Schaltern am SW 1 eingestellt ist. Bei Lieferung ist diese &HDE00. Der Unterroutinen Offset, ein Wert zwischen 1 und 30, im Normalfall 21, teilt der Schnittstelle eine Adresse zur Initialisierung zu. Die Controller-Priorität wird auf verschiedene Pegelwerte eingestellt. Pegel% = 0: HO80 Interface ist Systemcontroller. Pegel% = 1: HO80 Interface ist ein Aktiver Controller und Pegel% = 2: HO80 Interface ist ein Gerät. Mit diesen Werten wird bei einem Multicontrollerbetrieb festgelegt welcher Rechner der Systemcontroller ist. Ein Controller kann auch die Funktion eines Gerätes annehmen z.B. zur Meßdatenablage, oder als Subcontroller für bestimmte Aufgaben bei der Meßdatenerfassung als aktiver Controller. Nach dem Befehl CALL INIT mit Pegel% = 0 sendet die HO-80 Schnittstelle ein (IFC) Interface Clear zur Initialisierung aller am IEEE-BUS befindlichen Geräte. Für das Arbeiten mit BASIC-Interpreter empfiehlt es sich die mitgelieferten Basic-Treiber zu installieren. Sie dienen zur Geschwindigkeitssteigerung indem das ROM in den RAM-Bereich kopiert wird. (Beispiel Seite 14).

SENDEN und EMPFANGEN

Mit den Befehlen IWD und IRD wird der Daten-Transfer vorgenommen. Diesen Variablen wird ein Unterroutinen-Offset zugewiesen als direkte Adressen. Wobei der Syntax folgendes bedeutet: IWD = Interface Write Data (SENDEN) bzw. IRD = Interface Read Data (EMPFANGEN).(Beispiel Seite 14).

SERVICE REQUEST mit SERIELL POLL

Für die Abfrage des Service Request gibt es keinen direkten Interruptbefehl. Das Überwachen der Funktion SRQ muß mittels Direktabfrage der Adreßleitung am IC µPD7210C erfolgen. Das Programmbeispiel zeigt die SRQ Abfrage mit zwei Geräten HM 8112 mit unterschiedlich eingestellter Integrationszeit und Seriell Poll Auswertung. Nach der Initialisierung und den Geräte-Befehlen wird für beide Geräte Seriell Poll aktiviert. Danach wartet das Programm in Zeile 420 bis sich eines der Geräte mit SRQ meldet. In den Zeilen 450-540 wird nun die Auswertung des Seriell Poll vorgenommen um das entsprechende Gerät zu ermitteln, welches sich mit SRQ gemeldet hat. Nach dem Abfragebefehl

ISPL (Interface Seriell Poll) wird automatisch der Zustand zurückgesetzt, so daß eine erneute Abfrage möglich wird. Über die Statusvariablen in den Zeilen 580 und 640 wird selektiert und in der nachfolgenden Routine dann die Geräte bedient. Haben beide Geräte gleichzeitig SRQ gesendet, so werden die Geräte nacheinander bedient. Diese Routine ist geräteunabhängig und kann in ihrer Struktur auch auf andere Anwendungen übertragen werden. Hierbei ist zu beachten, daß die eingestellte Controlleradresse des IC uPD7210 (bei Lieferung eingestellt auf &H2B8) im Programm die gleiche Wertigkeit besitzt. (Beispiel Seite 15).

SEKUNDÄRADRESSE und direkt Kommandos
In diesem Programmteil zeigen wir Ihnen, wie ein nur als TALKER spezifiziertes Gerät, mittels der SEKUNDÄR - Adresse in der Lage ist, verschiedene Modi anzunehmen. Als Beispiel wurde hierfür das Oscilloscope HM 208 ausgewählt. (Beispiel Seite 15).

Initialisierung
Nach der Initialisierung Zeile 180-300 wird ein SEC Befehl mit CALL IWSD (Interface-Write-Sekundär-Data) an das angeschlossene Gerät in Zeile 400 gesendet. Nach erfolgter Datenübertragung, siehe Zeile 650-680 wird ein RESET an den IEEE-Bus mittels Sekundärkommando gesendet.

Befehlsstring direkt Kommandos
In einem Befehlsstring wird dem Controller mitgeteilt (MLA), daß der PC zum Listener wird, daß der Befehl an das Gerät mit der Nr. 6 (TALK 6) gesendet werden soll, und daß das angesprochene Gerät sich in den Betriebszustand 10 zu setzen hat (SEC 10). Diese Syntax gilt für alle Direkt-Kommandos, welche auf Seite 8 aufgeführt sind.

DATENBLOCK LESEN (Array transfer)
In den Zeilen 420 bis 500 folgen die OFFSET und Variablenangaben für den Block-Transfer. Diese Übertragungsart ist die schnellste. Die Daten werden im BINÄR-Format übergeben. Es erfolgen nur am Anfang und am Ende der Übertragung die HAND-SHAKE Routinen. Einen 2-KByte großen Datenblock zu übertragen dauert in der Regel nicht länger als 0,2 Sekunden. Mittels eines PEEK Kommandos können dann die Werte aus dem Speicher direkt gelesen werden. Siehe hierzu Zeile 600-630. (Beispiel Seite 16).

DATENBLOCK SCHREIBEN (Array transfer)
Das Senden eines Datenblocks im Binärformat gehört zu der schnellsten Übertragungsart. Sie erfordert genaue Kenntnisse über die zu übertragenen Daten. Da beim Speichern der Werte in einem Integer-Array die Werte im PC-Speicher als 2 mal 8 Bit Information abgelegt wird, können beim Übertragen Differenzen auftreten. Die Übertragung zum Beispiel des Wertes für "A" (65) hätte zur Folge, daß das erste Byte mit dem Wert 65 belegt ist. Das Zweite Byte jedoch 0 enthält. Nach der Übertragung eines Speicherbereiches hat nun jedes 2. Byte beim Empfänger den Wert 0. Es ist nun hier die Aufgabe des Programmierers die Werte lückenlos in den Speicher zu schreiben zum Beispiel mit POKE.

Das Programmbeispiel
Senden von einem Datenblock zeigt nach der Initialisierung eine Sekundär Adressen Zuweisung MTA (Computer wird Talker 'Sprecher') und gibt für das Gerät Nr. 6 eine LISTEN (Listner 'Hörer') Anweisung. Gerätebezogen kann zusätzlich für das als Hörer angesprochene Gerät noch ein Direktbefehl notwendig sein, welcher mit CALL IWDA übertragen werden muß. Nach der Ermittlung der Offsetadressen wird der Datenblock mit dem Befehl CALL IWDA (Interface write Data Array) gesendet.

Programmierbeispiel
Arraydaten generieren für die Zahlenwerte 1 bis 255. Gesamtgröße des Datenblocks 512 Byte.

```
100 DIM A(512), B%(256)
110 DEF SEG
120 FOR I = 1 TO 255    'Datenblock mit den Werten 1 und 255
130  A(I*2)   =   1    'füllen
140  A(I*2+1)  = 255
150 NEXT I
160 '
170 FOR I = 1 TO 512
180  IF INT(A(I*2+1)/128) = 1 THEN A% = 1 ELSE A% = 0
190  IF A% = 1 THEN B%(I) = ((A(I*2)-1 XOR 255) + ((A(I*2+1)
 -128) XOR 127) * 256) * -1 : GOTO 210
200  B%(I) = A(I*2) + A(I*2+1) * 256
210 NEXT I
220 '
230 'Datenblock auf Bildschirm ausgeben
240 '
250 ADR = VARPTR(B%(1))
260 '
270 FOR AUSLESE = ADR TO ADR + 512
280 PRINT PEEK(AUSLESE);
290 NEXT
```

Direktprogrammierung des IEEE-Buscontrollers

Der Bus-Controllerbaustein uPD7210C kann mittels INP und OUT Befehlen direkt angesprochen werden. Die dazu notwendigen Register-Adressen können Sie der Produktbeschreibung des Herstellers entnehmen.

Status Mitteilungen

Die von der Schnittstelle ausgegebenen Statusmitteilungen können bei der Fehlersuche sehr nützlich sein. Die Variable STATUS kann entsprechend der Übertragungsart folgende Werte besitzen.

Mode	Wert	Syntax
Allgemein	0	Kein Fehler
IWSD	1	Falscher Befehl
	2	END Kommando in Sende oder Empfangsstring
	4	END Kommando in Sende oder Empfangsstring
	8	Zeitüberschreitung
	16	Reihenfolge Kommandostring fehlerhaft
IRSD	2	Empfangen ohne definierten Empfänger
	8	Zeitüberschreitung
IWD	8	Zeitüberschreitung
ISPL	8	Zeitüberschreitung
IRD	8	Zeitüberschreitung
IWDA	2	Senden ohne definierten Sender
	8	Zeitüberschreitung
IRDA	2	Empfangen ohne definierten Empfänger
	8	Zeitüberschreitung
	32	Datenübertragungende EOI bei falscher ANZAHL%

Demoprogramm Init, S + E

```
100'-------------------------------------------------------
110'
120'Beispiel Senden und Empfang von Daten
130'Send = IWD und Empfang = IRD
140'
150'-------------------------------------------------------
160'
170 DEF SEG = &HDE00                    'Speicheradresse
180 INIT = 0 : IWD = 9 : IRD = 21       'Unterroutinen Offset
190 '
200 '
210 ADRESS% = 21                        'HO-80 Adresse
220 PEGEL% = 0                          'Systemcontroller
230 '
240 '
250 INSTRUMENT% = 7                     'Geräteadresse
260 '
270 '
280 CALL INIT (ADRESS%,PEGEL%)          'Schnittstelle aktivieren
290 '
300 BEFEHL$ = "VDR1T3"                  'Geräte Funktion HM 8112
310 '                                   Volt DC
320 '                                   Bereich 0,2V
330 '                                   Integrationszeit 1 Sek
340 '
350 CALL IWD (INSTRUMENT%,BEFEHL$,STATUS%)
360 '
370 '                                   sendet Funktionsbefehl zum Instrument
380 '
390 LESE$ = SPACE$(30)                  'erzeugen Leerstring
400 CALL IRD (LESE$,LANG%,INSTRUMENT%,STATUS%)
410 '
420 '                                   lese Meßwert Multimeter
430 '
440 '
450 PRINT LEFT$(LESE$,LANG%)            'Ausgabe Datensatz
```

Demoprogramm SRQ

```
100'-------------------------------------------------------
110'
120'Beispiel: SRQ und Seriell Poll mit zwei Geräten
130'
140'-------------------------------------------------------
150'
160 DEF SEG = &HDE00
170 INIT = 0 : IWD = 9 : IRD = 21  : ISPL = 12
180 ADRESS% = 21 : PEGEL% = 0
190 CALL INIT (ADRESS%,PEGEL%)
200 '
210 INSTRUMENT1% = 7                    Geräteadressen
220 INSTRUMENT2% = 8
230 S1$="VDR1T3L0S0Q1"                  'Gerätebefehle
240 S2$="VDR1T4L0S0Q1"
250                                     'Demo für 2 Stück HM 8112
```

```
260             'Meßbereich: 0.2 Volt DC
270             'Integrationszeit Gerät 1: 1 Sekunde
280             'Integrationszeit Gerät 2: 10 Sekunden
290             'kontinuierliches Messen
300             'mit Service Request
310 '
320 CALL IWD (INSTRUMENT1%,S1$,STATUS%)     'Sende
330                                          'Funktions-
340 CALL IWD (INSTRUMENT2%,S2$,STATUS%)     'befehl zu den
350                                          'Instrumenten
360'
370'
380 CALL ISPL (INSTRUMENT1%,pol%,STATUS%)   'aktivieren
390 CALL ISPL (INSTRUMENT2%,pol%,STATUS%)   'seriell Poll
400'
410'
420 IF (INP(&H2BA) AND &H40) = 0 THEN 420   'warten auf SRQ
430                                          'auf I/O Adresse
440 ST1 = 0 : ST2 = 0                        '&H2BA
450 '
460 CALL ISPL (INSTRUMENT1%,pola%,STATUS%)
470 i =  (pola% AND &H40 )
480 LOCATE 3,5 : PRINT "poll 1 = "; pola%    'Status poll 1
490 IF pola% > 0 THEN ST1 = 1
500 '
510 CALL ISPL (INSTRUMENT2%,polb%,STATUS%)
520 i =  (polb% AND &H40 )
530 LOCATE 4,5 : PRINT "poll 2 = "; polb%    'Status poll 2
540 IF polb% > 0  THEN ST2 = 1
550 '
560 '
570 '
580 IF ST1 = 0 THEN 640
590 L$ = SPACE$(40)
600 CALL IRD (L$,LANG%,INSTRUMENT1%,STATUS%)
610 LOCATE 6,5 : PRINT "Gerät 1    " L$    Meßwert
630 '
640 IF ST2 = 0 THEN 420
650 L$ = SPACE$(40)
660 CALL IRD (L$,LANG%,INSTRUMENT2%,STATUS%)
670 LOCATE 8,5 : PRINT "Gerät 2    " L$    Meßwert
690 GOTO 420
```

Demoprogramm SEC und IRDA

```
100'-------------------------------------------------------
110'Beispiel: Betriebsart Auslese-Mode durch Sekundäradresse
120'aktivieren und Daten einlesen im Binärmode (Array transfer).
130'Ein Demoprogramm zum HM 208.
140'-------------------------------------------------------
150'
160 DIM DA(2050), BLOCK(2050)                'Dimensionierung für
170 '                                        Block auslesen.
180 '
190 DEF SEG = &HC400                         'Speicheradresse
200 INIT = 0: IWD = 9: IRD = 21              'Unterroutinen Offset
```

```
210 IWSD = 3: IRSD = 6
220 '
230 ADRESS% = 21                          'PC Controller Adresse 21
240 PEGEL% = 0                            'Systemcontroller
250 '
260 INSTRUMENT% = 6                       'Geräteadresse TALKER
270 '
280 '
290 CALL INIT(ADRESS%, PEGEL%)            'Schnittstelle aktivieren
300 '
310 '————————Direkt Kommando SEKUNDÄR ADRESSE ausführen————————
320 '
330 DEF SEG = &HC400
340 '
350 BEFEHL$ = "MLA TALK 6 SEC 10"         'SEC-Befehl an Geräteadresse 6
360 '                                     Sekundäradresse 10 aktivieren.
370 CALL IWSD(BEFEHL$, STATUS%)           'Eingabe der Sekundäradresse
380 '                                     zum Daten - Block auslesen.
390 '
400 '————————————————Datenblock lesen————————————————
410 '
420 IRDA = 203
430 '
440 ANZAHL% = 2048                        'Anzahl der zu übertragenen
450 '                                     Datenbyts.
460 SEGMENT% = -1
470 '
480 OFFSET% = VARPTR(BLOCK(0))            'Hauptspeicheradresse
490 '                                     erstes Datenbyte
500 '
510 CALL IRDA(SEGMENT%, OFFSET%, ANZAHL%, LANG%, STATUS%)
520 '                                     Befehl zum Übertragen eines
530 '                                     Datenblocks
540 '
550 '————————————————Datenausgabe Block lesen————————————
560 '
570 DEF SEG = &HC400: DEF SEG             'Rücksetzen Speicheradresse
580 '                                     'auf Basic Datensegment
590 '
600 FOR I = 1 TO 2048                     'Ausgabe Daten
610 DA(I) = PEEK(OFFSET% + I)
620 PRINT DA(I);
630 NEXT I
640 '
650 '————————————————Reset IEEE - Bus————————————————
660 DEF SEG = &HC400
670 BEFEHL$ = "UNL UNT DCL"
680 CALL IWSD(BEFEHL$, STATUS%)
690 END
```

Demoprogramm IWDA

```
100 '————————————————————————————————————————
110 'Beispiel: Betriebsart Senden von einem Datenblock
120 ' binärer Daten (Array transfer).
130 '————————————————————————————————————————
```

```
140 '
150 DIM BLOCK% (1024)                    'Dimensionierung für
160 '                                    Block senden
180 DEF SEG=&HDE00                       'Speicheradresse
190 INIT = 0 : IWD = 9 : IRD = 21        'Unterroutinen Offset
200 IWSD = 3 : IRSD = 6
210 '
220 ADRESS%=21                           'PC Controller Adresse 21
230 PEGEL%=0                             'Systemcontroller
240 '
250 INSTRUMENT%=6                        'Geräteadresse LISTNER
260 '
270 '
280 CALL INIT(ADRESS%,PEGEL%)            'Schnittstelle aktivieren
290 '
300 '----------Direkt Kommando ausführen----------
310 '
320 DEF SEG=&HDE00
330 '
340 BEFEHL$= "MTA LISTEN 6"              'Aktivieren Listener
350 '
360 CALL IWSD (BEFEHL$,STATUS%)          'Eingabe der Sekundäradresse
370 '                                    zum Daten - Block schreiben
390 '----------Datenblock erzeugen----------
400 FOR I = 1 TO 1024 : BLOCK%(I) = 68 : NEXT
410 '       Datenblock füllen mit dem ASCII-Wert von A
420 '       Hier ist die Speicherverwaltung des Computers zu beachten.
430 '       Integerwerte belegen 2 Byte im RAM. Ist der Wert 8 Bit
440 '       oder kleiner, wird zwischen jedem übertragenem Datenbyte eine
450 '       Null mit übertragen.
460 '----------Datenblock schreiben----------
470 '
480 IWDA = 200
490 '
500 ANZAHL%=1024                         'Anzahl der zu übertragenen
510 '                                    Datenbyts.
520 '
530 SEGMENT% = -1 : OFFSET% = 0 : OFFSET = 0 : STATUS% = 0
540 '
550 'OFFSET% = VARPTR (BLOCK%(1))        'Hauptspeicheradresse
560 '                erstes Datenbyte
570 '
580 EOI% = 1  'Endzeichen letztes Datenbyte EOI = 0 nein
590 '                    EOI = 1 ja
610 OFFSET% = VARPTR (BLOCK%(1))         'Hauptspeicheradresse
620 CALL IWDA (SEGMENT%,OFFSET%,ANZAHL%,EOI%,STATUS%)
630 '                                    Befehl zum Übertragen eines
640 '                                    Datenblocks
650 '
660 '----------RESET IEEE - Bus----------
670 DEF SEG = &HDE00
680 BEFEHL$ = "UNL UNT DCL"
690 CALL IWSD (BEFEHL$,STATUS%)
700 END
```

UNIT HO80_1;

INTERFACE

VAR

{Die beiden Variablen IEEE_card und IEEE_SEG dürfen nur gelesen werden und sind nicht zu verändern!}

IEEE_card: boolean; { true : IEEE-Karte ist installiert }
{ false : IEEE-Karte ist nicht installiert }

IEEE_SEG : word; { enthält Segment-Adresse der IEEE-Kartenadr. }
{ wird automatisch bestimmt, nicht verändern ! }

{ Es folgen vordefinierte Variablen für die Verwendung der Proceduren. }

Pegel,
Status,
Instrument,
Pol,
Anzahl,
Lang,
Eoi,
Adress : integer;
Kommando,
Befehl : string;

{ Folgende Pointer sind Einsprungadressen für Assemblerroutinen }
{ nicht verändern ! }

ADR_INIT, ADR_IWSD, ADR_IRSD, ADR_IWD, ADR_IRD,
ADR_ISPL, ADR_IPPL, ADR_IWDA, ADR_IRDA, ADR_DEVICE : pointer;

Initialisierung

PROCEDURE INIT

(VAR Adress, Pegel : integer);
{ Adress = PC Geräte Adresse, z.B. 21 [0..30 }
{ Pegel = Controllereinstellung }
{ 0 : System - Controller }
{ 1 : aktiver - Controller }
{ 2 : Gerät; kein Controller }

Übermittlung von Gerätenachrichten, bzw. Datenworten

PROCEDURE IWD

(VAR Instrument : integer; VAR Befehl : string; VAR Status : integer);

{ sende Gerätenachricht, bzw. Datenwort }
{ Instrument = Adresse des Empfangsgerätes }
{ Befehl = Befehlsstring, Gerätenachricht }
{ Status = Übermittlungsstatus }

PROCEDURE IRD

```
(VAR Befehl : string; VAR Instrument : integer; VAR Status
 :integer);

{   empfange Gerätenachricht, bzw. Datenwort          }
{   Befehl      = Befehlsstring, Gerätenachricht      }
{   Instrument  = Adresse des Sendegerätes            }
{   Status      = Übermittlungsstatus                 }
```

Übermittlung von Schnittstellen - und Gerätenachrichten, bzw. Datenworten

PROCEDURE IRSD

```
( VAR Kommando   : string;  VAR Status   : integer);

{ empfange Schnittstellen- und Gerätenachricht, bzw. Datenwort      }
{ Kommando=Befehlsstring, Schnittstellen- bzw. Gerätenachricht      }
{ Status   = Übermittlungsstatus                                    }
```

PROCEDURE IWSD

```
( VAR Kommando   : string;  VAR Status   : integer);

{ sende Schnittstellen- und Gerätenachricht, bzw. Datenwort         }
{ Kommando=Befehlsstring, Schnittstellen- bzw. Gerätenachricht      }
{ Status = Übermittlungsstatus                                      }
```

Übermittlung von Datenarrays

```
{ Vor Aufruf der Proceduren IRDA und IWDA ist einem Pointer   }
{ (PtrDA) die Adresse des Arrays zu übergeben                 }
{  PtrDA := addr (DA);                                        }
{  wobei DA das Datenarray ist, zB DA : array [0..1023] of byte }
```

PROCEDURE IRDA

```
( VAR PtrDA    : pointer;  VAR Anzahl, Lang, Status : integer);

{ empfange Datenarray                          }
{ PtrDA     = Zeiger auf das Datenarray        }
{ Anzahl = Grösse des Datenarray               }
{ Lang      = übermittelte Datensatzlänge      }
{ Status    = Übermittlungsstatus              }
```

PROCEDURE IWDA

```
( VAR ptrDA    : pointer;  VAR Anzahl, Eoi, Status : integer);

{   sende Datenarray}
{   PtrDA       = Zeiger auf das Datenarray          }
{   Anzahl      = Anzahl der zu übertragenden Daten  }
{   Eoi         = Übertragung von EOI                }
{               0 : nein                             }
{               1 : ja                               }
{   status      = Übermittlungsstatus                }
```

```
IWD (Instrument, Befehl, Status);        { sendet Befehl zum
                                           Instrument          }

FILLCHAR (Befehl, 80, ' '); Befehl[0] := #80;    { erzeugt
                                                   Leerstring }

IRD (Befehl, Instrument, Status);        { lese Meßwert Multimeter   }

writeln (Befehl);                        { Ausgabe Datensatz         }

END.                                     { IEEEPAS1                  }
```

Demoprogramm Turbopascal Sekundäradressen und Blocktransfer

```
PROGRAM IEEEPAS2;

{ DEMO-PROGRAMM IEEEPAS2                                              }
{ BEISPIELPROGRAMM FUER INITIALISIERUNG, SENDUNG VON                  }
{ SEKUNDÄRADRESSEN UND DATENARRAYTRANSFER                             }
{                                                                     }
{ UNIT H080_1                                                         }
{ STELLT FÜR TURBO-PASCAL DIE VERWENDETEN PROCEDUREN ZUR              }
{ VERFÜGUNG                                                           }
{ DIE KARTENADRESSE DES HO80 WIRD AUTOMATISCH FESTGESTELLT !          }
{ ALLGEMEINE BEZEICHNER SIND VORDEFINIERT.- SIEHE PROGRAMM-           }
{ BESCHREIBUNG TURBOPASCAL - UNITS                                    }

USES
HO80_1;
VAR
 I : INTEGER;

DA : ARRAY [0..2050] OF BYTE;          {DataArray                     }
PtrDA : pointer;      { Zeiger für array-Transfer IRDA,IWDA           }

 BEGIN              { IEEEPAS2                                        }
 IF not IEEE_card then HALT;    { Ausführung beenden falls            }
                                { kein IEEE-Bus vohanden              }
 PtrDA := addr (DA);            { Zuordnung Datenarray zu             }
                                { Zeiger-Var.                         }

{ ———————— Initialisierung des IEEE-Interface ———————— }
 Adress := 21;                  { HO80 Adresse                        }
 Pegel  := 0;                   { Systemcontroller                    }
 INIT (Adress, Pegel);          { Schnittstelle aktivieren            }

{— Direkt Kommando SEKUNDAER ADRESSE AUSFUEHREN —          }

 Kommando := 'MLA TALK 6 SEC 10';  { SEC-Kommando an      }
                                   { Geräteadresse 6      }
                                   { Sekundäradresse 10   }
                                   { aktivieren           }
 IWSD (Kommando, STATUS);          { sendet Kommando      }
```

```
{———————————— Datenblock lesen ————————————}

 ANZAHL := 2046;                          { Anzahl der array-Grösse      }
 IRDA (PtrDA, ANZAHL, LANG, STATUS);      { Befehl zum Übertragen        }
                                          { von einem array              }
{———————————— Datenblock anzeigen ————————————}
 FOR I := 1 TO LANG DO
  BEGIN
  WRITE (DA[I]:4);
  IF I MOD 16 = 0 THEN WRITELN;
  END;
{———————————— Reset IEEE - Bus ————————————}
 Kommando := 'UNL UNT DCL';
 IWSD (Kommando, STATUS);
 END.  { IEEEPAS2 }
```

Programmbeschreibung C Bezeichnerliste: "MC.H"

```
 INCLUDE-Datei für  "QuickC 1.01"
 ( geschütztes Warenzeichen von Microsoft)
 INCLUDE-Datei für  "TURBO C"
 ( geschütztes Warenzeichen von Borland International )
 Bezeichnerliste für Assembler-Funktionen
 zur Unterstützung der HAMEG Interface-Karte HO80-IEEE488

 in den Beispielen aufgeführt Bezeichner
 sind folgendermaßen zu definieren :

 int      controller,
          my_adress,
          status,
          instument,
          anz,
          eoi,
          lang,
          pol;

 char     far *nachricht,
          far *antwort;
*/

#define ho80_seg HO80_SEG
#define ieee_se IEEE_SE
#define init INIT
#define iwd IWD
#define ird IRD
#define iwsd IWSD
#define irsd IRSD
#define iwda IWDA
#define irda IRDA
#define ispl ISPL
#define ippl IPPL

extern unsigned int  HO80_SEG;
```

```
/* HO80_SEG = 0xC400;                                          */
extern int IEEE_SE (unsigned int);
/* if (IEEE_SE (HO80_SEG) != 0) exit (1);                      */

extern void INIT(int far *,int far *);
/* INIT (&controller, &my_adress);                             */

extern void IWD(int far *,char far *,int far *);
/* IWD (&status, nachricht, &instrument);                      */

extern void IRD(int far *,int far *,int far *,char far *);
/* IRD (&status, &instrument, &anz, antwort);                  */

extern void IWSD (int far *, char far *);
/* IWSD (&status, nachricht);                                  */

extern void IRSD (int far *, int far *, char far *);
/*    IRSD (&status, &anz, antwort);                           */

extern void IWDA(int far *,int far *,int far *,int far *,int far *);
/* IWDA (&status, &eoi, &lang, data, &dummy)                   */

extern void IRDA(int far *,int far *,int far *,int far *,int far *);
/* IRDA (&status, &anz, &lang, data, &dummy)                   */

extern void ISPL(int far *,int far *,int far *);
/* ISPL (&status, &pol, &instrument);                          */

extern void IPPL(int far *);
/* IPPL (%pol);                                                */

/*
Die Funktionensbeschreibung ist dem Handbuch aus der Befehlsliste
zu entnehmen.
Hier werden nur Abweichungen zu diesen Erklärungen aufgeführt.

HO80_SEG
Speicheradresse der HO80 Interfacekarte
Die Adresse ist auf C400H vordefiniert.
Wird eine andere Adresse verwendet, so ist diese
HO80_SEG zuzuweisen. (z.B.: HO80_SEG = 0xC000 )

IEEE_SE
Die Funktion testet ob an der übergebenen Segmentadresse eine HO80 Interfacekarte installiert ist,
und übergibt die Segmentadresse ho80_seg. Diese Funktion sollte vor IEEE-Routinen aufgerufen
werden.
(Siehe Programm HM8112.C )
Ist die Segmentadresse unbekannt, kann sie folgendermaßen ermittelt werden:
    HO80_SEG = 0x0000;
    while ((ieee_se (HO80_SEG)) && (HO80_SEG < 0xFE00))
    { HO80_SEG += 0x0200; }IWDA, IRDA
Die Funktion entspricht der Beschreibung im Handbuch; die Segmentübergabe ist jedoch nur aus
Kompatibilitätsgründen aufgeführt und kann ein beliebiger Zeiger sein.
*/
```

Demoprogramm MS-C

```
/*
Beispielprogramm für Interfacekarte HO80 IEEE unter Quick C 1.01
*/

/*
Je nach Speichermodell sind unterschiedliche Module beim Kompilieren einzubinden :
                SMALL  : CSCOMMON.OBJ
                MEDIUM : CMCOMMON.OBJ
                LARGE  : CLCOMMON.OBJ
*/

#include "MC.H"
main ()
{
 int my_adress, system_controller;
 int instrument;
 int status, pol;
 int segdata;
 int anz, eoi, lang;
 unsigned char data [2050];
 int bst;
 char recv[80];
 char *cmd0 = "MLA TALK 8";
 char *cmd1 = "MLA TALK 8 SEC 1";
 char *cmd2 = "MLA TALK 8 SEC 4";
 char *cmd3 = "MLA TALK 8 SEC 10";
 char *cmd4 = "MLA TALK 8 SEC 15";
 int ende;

 HO80_SEG = 0x0000;
 while ((ieee_se (HO80_SEG)) && (HO80_SEG < 0xFE00))
 {
  HO80_SEG += 0x0200;
 }

 if (ieee_se(HO80_SEG))
 {
  printf ("HO80-IEEE nicht gefunden");
  exit (1);
 }
 else
 {
  printf ("HO80-IEEE auf Adresse %X installiert,\n\n\n",HO80_SEG);
 }
 my_adress = 21; system_controller = 0;
 instrument = 8;
 init (&system_controller,&my_adress);
 ende = 0;
 while (!ende)
 {
  printf ("Versions-Nummer                          (0)\n");
  printf ("Datenbyte von Scope-Speicher lesen       (1)\n");
  printf ("Trigger-Rücksetzung und Statusmeldung    (2)\n");
  printf ("Scope-Speicher auslesen                  (3)\n");
  printf ("Statusmeldung                            (4)\n\n");
  printf ("Programm beenden                         (Q)\n\n\n");
  bst = getche();
```

```
printf ("\n");
 switch (bst)
 {
  case '0':
  /*  IRD  -  Interface Read data              */
   strcpy (recv,"                              ");
   ird (&status,&instrument,&anz,recv);
   printf ("empfangene Daten = '%s', Anzahl = %d\n",recv,anz);
   printf ("Status     = %X\n\n\n",status);
   break;
  case '1':
   /*  IWSD  -  Interface Write Secundary Data   */
   printf ("gesendete Daten : '%s'\n",cmd1);
   iwsd (&status,cmd1);
   printf ("Status = %X\n\n",status);
   /*  IRDA  -  Interface Read Data Array        */
   lang = 1;
   irda (&status,&anz,&lang,data,&segdata);
   printf ("Anzahl der empf. Daten : %d (%d)\n",anz,data[0]);
   printf ("Status     = %X\n\n\n",status);
   break;
  case '2':
   /*  IWSD  -  Interface Write Secundary Data   */
   printf ("gesendete Daten : '%s'\n",cmd2);
   iwsd (&status,cmd2);
   printf ("Status = %X\n\n",status);
   /*  IRSD  -  Interface Read Secundary Data    */
   strcpy (recv,"                              ");
   irsd (&status,&anz,recv);
   printf ("empfangene Daten = '%s', Anzahl = %d\n",recv,anz);
   printf ("Status     = %X\n\n\n",status);
   break;
  case '3':
   /*  IWSD  -  Interface Write Secundary Data   */
   printf ("gesendete Daten : '%s'\n",cmd3);
   iwsd (&status,cmd3);
   printf ("Status = %X\n\n",status);
   /*  IRDA  -  Interface Read Data Array        */
   lang = 2049;
   irda (&status,&anz,&lang,data,&segdata);
   printf ("Anzahl der empf. Daten : %d\n",anz);
   printf ("Status     = %X\n\n\n",status);
   break;
  case '4':
   /*  IWSD  -  Interface Write Secundary Data   */
   printf ("gesendete Daten : '%s'\n",cmd4);
   iwsd (&status,cmd4);
   printf ("Status = %X\n\n",status);
   /*  IRSD  -  Interface Read Secundary Data    */
   strcpy (recv,"                              ");
   irsd (&status,&anz,recv);
   printf ("empfangene Daten = '%s', anz = %d\n",recv,anz);
   printf ("Status     = %X\n\n\n",status);
   break;
  case 'Q': ende = 1;
 }
 }
 exit (0);
}
```

ASCII & IEEE (GPIB) CODE CHART

B7 B6 B5 / BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 0 NUL 0	20 10 DLE 16	40 20 LA 0 SP 32	60 30 LA 16 0 48	100 40 TA 0 @ 64	120 50 TA 16 P 80	140 60 SA 0 ` 96	160 70 SA 16 p 112
0 0 0 1	1 1 GTL SOH 1	21 11 LLO DC1 17	41 21 LA 1 ! 33	61 31 LA 17 1 49	101 41 TA 1 A 65	121 51 TA 17 Q 81	141 61 SA 1 a 97	161 71 SA 17 q 113
0 0 1 0	2 2 STX 2	22 12 DC2 18	42 22 LA 2 " 34	62 32 LA 18 2 50	102 42 TA 2 B 66	122 52 TA 18 R 82	142 62 SA 2 b 98	162 72 SA 18 r 114
0 0 1 1	3 3 ETX 3	23 13 DC3 19	43 23 LA 3 # 35	63 33 LA 19 3 51	103 43 TA 3 C 67	123 53 TA 19 S 83	143 63 SA 3 c 99	163 73 SA 19 s 115
0 1 0 0	4 SDC 4 EOT 4	24 DCL 14 DC4 20	44 24 LA 4 $ 36	64 34 LA 20 4 52	104 44 TA 4 D 68	124 54 TA 20 T 84	144 64 SA 4 d 100	164 74 SA 20 t 116
0 1 0 1	5 PPC 5 ENQ 5	25 PPU 15 NAK 21	45 25 LA 5 % 37	65 35 LA 21 5 53	105 45 TA 5 E 69	125 55 TA 21 U 85	145 65 SA 5 e 101	165 75 SA 21 u 117
0 1 1 0	6 6 ACK 6	26 16 SYN 22	46 26 LA 6 & 38	66 36 LA 22 6 54	106 46 TA 6 F 70	126 56 TA 22 V 86	146 66 SA 6 f 102	166 76 SA 22 v 118
0 1 1 1	7 7 BEL 7	27 17 ETB 23	47 27 LA 7 ' 39	67 37 LA 23 7 55	107 47 TA 7 G 71	127 57 TA 23 W 87	147 67 SA 7 g 103	167 77 SA 23 w 119
1 0 0 0	10 GET 8 BS 8	30 SPE 18 CAN 24	50 28 LA 8 (40	70 38 LA 24 8 56	110 48 TA 8 H 72	130 58 TA 24 X 88	150 68 SA 8 h 104	170 78 SA 24 x 120
1 0 0 1	11 TCT 9 HT 9	31 SPD 19 EM 25	51 29 LA 9) 41	71 39 LA 25 9 57	111 49 TA 9 I 73	131 59 TA 25 Y 89	151 69 SA 9 i 105	171 79 SA 25 y 121
1 0 1 0	12 A LF 10	32 1A SUB 26	52 2A LA 10 * 42	72 3A LA 26 : 58	112 4A TA 10 J 74	132 5A TA 26 Z 90	152 6A SA 10 j 106	172 7A SA 26 z 122
1 0 1 1	13 B VT 11	33 1B ESC 27	53 2B LA 11 + 43	73 3B LA 27 ; 59	113 4B TA 11 K 75	133 5B TA 27 [91	153 6B SA 11 k 107	173 7B SA 27 { 123
1 1 0 0	14 C FF 12	34 1C FS 28	54 2C LA 12 , 44	74 3C LA 28 < 60	114 4C TA 12 L 76	134 5C TA 28 \ 92	154 6C SA 12 l 108	174 7C SA 28 | 124
1 1 0 1	15 D CR 13	35 1D GS 29	55 2D LA 13 - 45	75 3D LA 29 = 61	115 4D TA 13 M 77	135 5D TA 29] 93	155 6D SA 13 m 109	175 7D SA 29 } 125
1 1 1 0	16 E SO 14	36 1E RS 30	56 2E LA 14 . 46	76 3E LA 30 > 62	116 4E TA 14 N 78	136 5E TA 30 ^ 94	156 6E SA 14 n 110	176 7E SA 30 ~ 126
1 1 1 1	17 F SI 15	37 1F US 31	57 2F LA 15 / 47	77 3F UNL ? 63	117 4F TA 15 O 79	137 5F UNT _ 95	157 6F SA 15 o 111	177 7F SA 31 RUBOUT (DEL) 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY ASCII character

octal 25 PPU — GPIB code

NAK

hex 15 21 — decimal