LAB. 08 (http://diana.uceh.ualg.pt/Inst/lab08.pdf)
Interface IEEE488 (GPIB)

Material

computador interface CEC 488 (GPIB) e subrotinas em C Osciloscópio digital Tektronix TDS210

Objectivos

Pretende-se neste trabalho familiarizar o aluno com a interface IEEE488 e a sua aplicação em aquisição de dados num ambiente laboratorial

Introdução

0 "bus" IEEE-488, também designado por GPIB ("general-purpose interface bus"), é uma generalização do "bus" HP-IB desenvolvido pela companhia Hewlett-Packard. É um "bus" especialmente concebido para a transferência de dados entre instrumentos.

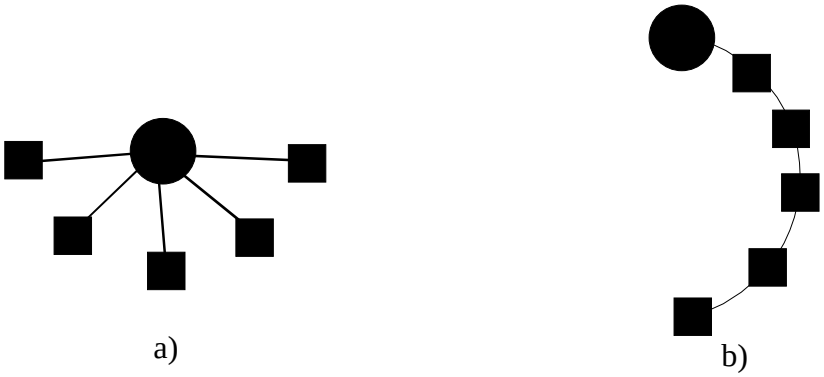

Figura 1: Configurações típicas do "bus" GPIB: a) estrela, b) linear.

Apresenta-se, na Figura 1, configurações típicas do "bus" GPIB. Várias configurações são possíveis: em estrela, linear, ou uma combinação dos dois modos.

São transmitidos caracteres, de 8 bits de comprimento, em série ("byte serial, bit parallel") no "bus". A taxa de transmissão máxima é 1 MB/s (250 kB/s típico). São permitidos, no máximo 15 aparelhos no "bus", num comprimento total de 20 metros.

Existem no "bus" instrumentos com funções e capacidades distintas: controladores (só um pode estar activo de cada vez), locutores ("talkers"), e ouvintes ("listeners").

Para não haver perda de informação, sé um locutor pode estar activo de cada vez. É possível a um instrumento chamar a atenção do controlador para ser atendido ("service request").

Cada instrumento tem um endereço distinto, de forma que a comunicação entre instrumentos possa ser feita sem haver colisões. Normalmente, no painel de trás do aparelho, ou noutro local de fácil acesso, existe um conjunto de cinco interruptores que permitem definir o endereço principal do instrumento. Os interruptores definem um endereço codificado em binário, que pode tomar um valor entre 0 e 30 (decimal).

É importante um conhecimento dos comandos do "bus" para compreender o seu funcionamento. Na eventualidade de não existirem disponíveis rotinas (um "device driver") para a interface de controlo do "bus", este pode ser controlado directamente utilizando os comandos.

Interface Message	Description	Decimal Value
PCG	*Primary Command Group*	
GTL	Go to Local	1
SDC	Selected Device Clear	4
PPC	Parallel Poll Configure	5
GET	Group Execute Trigger	8
TCT	Take Control	9
LLO	Local Lockout	17
DCL	Device Clear	20
PPU	Parallel Poll Unconfigure	21
SPE	Serial Poll Enable	24
SPD	Serial Poll Disable	25
LAG	*Listen Address Group*	
	Listen Address 0 through 30	32 – 62
	Unlisten	63
TAG	*Talk Address Group*	
	Talk Address 0 through 30	64 – 94
	Untalk	95
SCG	*Secondary Address Command Group*	
	Secondary Commands 0 through 30	96 – 126

Tabela 1: Tabela de comandos do "bus" GPIB

Mostra-se na Tabela 5 todos os comandos do "bus". Estes dividem-se em quatro grupos: grupo de comandos primários, grupo de endereços de ouvintes, grupo de endereços de locutores, e grupo de endereços secundários. O significado dos comandos primários é na sua maioria imediato.

A maior parte dos comandos primários necessita do envio prévio de um ou mais comandos do grupo de endereços de ouvintes para seleccionar quais os instrumentos que devem responder a estes comandos. Note-se ainda que em cada transmissão de dados, apenas se pode enviar previamente um comando do grupo de endereços de locutores, ou haverá confusão no "bus".

Estas breves linhas servem apenas de introdução à interface IEEE488, recomendando-se vivamente a leitura do Apêndice A e B, para a realização do trabalho.

Montagem Experimental

Neste trabalho pretende-se controlar a partir de um computador um osciloscópio digital Tektronix modelo TDS210.

O hardware não pode ser mais simples: basta ligar com um cabo GPIB o osciloscópio ao computador. O software é um pouco mais complicado.

O programa é realizado em Borland C (versão DOS). O compilador é activado com o comando

```
c:\>bc
```

Criar um novo projecto e incluir (na janela *project*) o nome do programa (*nome_do_programa.c*) e a livraria (*c:\ieee488\ieee-cpp.lib*).

Em *Options>Directories…*

indicar o directório de trabalho (*c:\users\a999999*) e os directórios onde se encontra as livrarias (*c:\borlandc\lib*) e os headers (*c:\borlandc\include*)

Para começar o programa de controlo pode ser tão simples como

```
/*
  Example 1: use of SEND & ENTER to communicate
  with an instrument (Tektronix TDS210 oscilloscope)
*/
#include "c:\ieee488\ieee-cpp.h"
#include <stdio.h>
#include <string.h>

#define TDS210 7

main ()

{
        int status, length;
        char s[40], r[4000];
        initialize(21,0);
        scanf("%s",&s) ;
        send (TDS210,s,&status);
        enter(r,4000,&length,TDS210,&status);
        printf("%d\t%s",length,r);
        return(0);

}
```

Alguns comandos que o osciloscópio aceita

curve[?]	exemplo: curve?
horizontal:scale [?, value]	exemplo: horizontal:scale 500e-6
ch<x>:scale [?, value]	exemplo: ch1:scale?

(quando o comando termina em "?" o osciloscópio responde…)

Algumas subrotinas disponíveis na livraria ieee-cpp.lib

INITIALIZE - inicializar a interface
SEND - enviar mensagem
ENTER - receber mensagem
SPOLL - serial poll
PPOLL - paralell poll
TRANSMIT - enviar mensagem (baixo nível)
RECEIVE - receber mensagem (baixo nível)
SRQ - service request

Pretende-se realizar um programa em C que

- configure remotamente o osciloscópio com
  escala vertical no canal 1: 2V/divisão
  escala temporal (horizontal): 250 μs/divisão
- faça a aquisição de um sinal, por exemplo o próprio sinal de calibração que existe no osciloscópio (é uma onda quadrada)

O programa em C tem que ser realizado utilizando as subrotinas de baixo nível TRANSMIT e RECEIVE.

O objectivo final é obter no computador sob a forma de tabela com duas colunas (tempo, sinal digitalizado) um *screen dump* do canal 1 do osciloscópio.

Para a realização deste trabalho é fundamental consultar os manuais de referência do osciloscópio TDS210 e da interface CEC 488!

NÃO LEVAR ESTES MANUAIS PARA FORA DO LABORATÓRIO DE INSTRUMENTAÇÃO

APÊNDICE A - Tutorial sobre o "Bus" IEEE488

Características gerais

0 standard IEEE 488 definia inicialmente as características mecânicas, eléctricas e funcionais de uma interface digital, de uso geral, para comunicação a curta distância entre instrumentos. Uma revisão do standard (IEEE 488.2) define ainda características funcionais mínimas das interfaces, formatos dos dados, protocolo das mensagens, e comandos comuns. Nesta secção, faz-se uma apresentação do "bus" GPIB, focando essencialmente as características inicialmente definidas pelo standard IEEE 488.

As características gerais do "bus" são apresentadas na Tabela 2

Data Rates 250 KB per second typical 1 MB per second maximum *Number of Devices on Bus* 15 devices maximum (electrical limit) 8 devices typical (parallel poll limit) *Bus Length* 20 m maximum 2 m per device typical *Bus Structure* Byte oriented 24 line bus 8 bits for commands 8 bits for data 8 bits for grounding *Interrupt Driven* Serial poll (slower response) Parallel poll (fast response) *Bus Hierarchy* Microcomputer/controller Talkers and listeners Only single active controller or talker allowed System may consist of talker and listeners only *Communications* At any time only one talker may be active Multiple listeners are allowed

Tabela 2: Especificações gerais do "bus" GPIB

"Hardware" do "bus"

Apresenta-se na Figura 2 um conector standard para ligação de instrumentos ao "bus". Um cabo de ligação tem normalmente terminações macho e fêmea na mesma extremidade, o que permite uma grande variedade de padrões de ligação, desde uma configuração linear ("daisy chain"), até uma configuração em estrela. Não há degradação no funcionamento do "bus" desde que se obedeçam à seguinte restrição: comprimento total máximo 20 metros, ou 2 metros por aparelho. A distância entre aparelhos não é crítica, desde que se tenha em consideração a restrição acima mencionada.

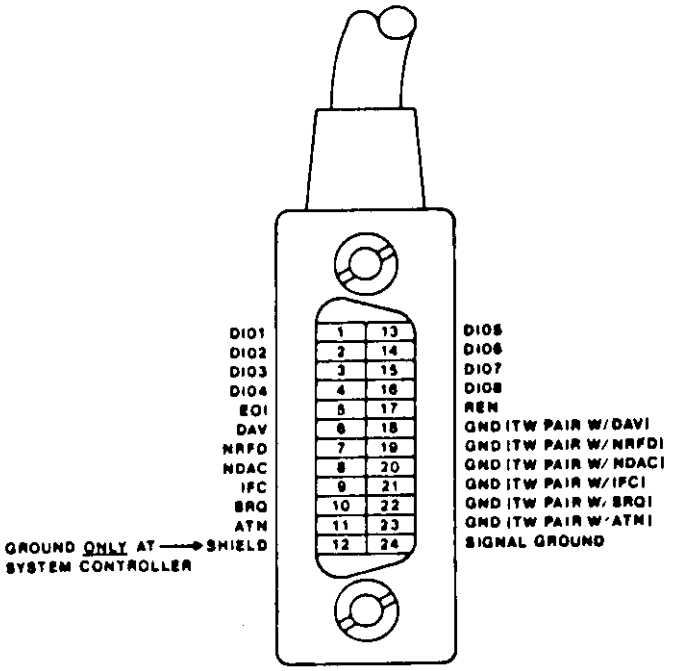

Figura 2: Conector standard GPIB

Cada conector e cabo de ligação tem um total de 24 linhas, onde oito são linhas de controle, oito são linhas de dados, e as restantes oito são linhas de terra. O standard GPIB utiliza lógica invertida, onde 0 TTL (<0.8V) representa um valor lógico verdadeiro e 1 TTL (>2.0V) representa um valor lógico falso.

Apresenta-se Tabela 3 algumas características eléctricas do "bus".

General.

The relation between logic and voltage levels is:

Logic Level	Voltage Level
0 (False)	≥ +2.0V (High)
1 (True)	≤ +0.8V (Low)

Driver Types

Open Collector Only	Open Collector or *Tristate
SRQ, NRFD, NDAC	ATN, IFC, REN, EOI, DAV
DIO1-8 (Parallel Poll devices)	DIO1-8 (non-Parallel Poll devices)

*Tristate useful to reach data rates above 250,000 bytes/sec. Tristate is disabled during parallel poll.

Driver Specifications

V_{OL} < +0.5V @ 48 ma continuous sink (tristate or open collector)
V_{OH} ≥2.4V @ 5.2 ma source (tristate)
see DC Load Line Graph (open collector)

Receiver Specifications

Preferred (Schmitt-type)	Allowed (non-Schmitt-type)
$V_{IL} = V_{tneg} \leq +0.8V$	$V_{IL} \leq +0.8V$
$V_{IH} = V_{tpos} \geq +2.0V$	$V_{IH} \geq +2.0V$
Hysteresis: $V_{tpos} - V_{tneg} \geq +0.4V$	

Tabela 3: Características eléctricas do "bus"

Endereçamento de instrumentos

Cada instrumento ligado ao "bus" tem um endereço distinto, de forma que a comunicação entre instrumentos possa ser feita sem haver colisões. Normalmente, no painel de trás do aparelho, ou noutro local de fácil acesso, existe um conjunto de cinco interruptores que permitem definir o endereço principal do instrumento. Os interruptores definem um endereço codificado em binário, que pode tomar um valor entre 0 e 30 (decimal). Podem existir mais interruptores, e neste caso os restantes definem o modo de operação do aparelho, por exemplo, sé capacidade para receber ("listen only") ou enviar ("talk only") mensagens, ou também ainda outras funções.

Existem instrumentos com capacidade de endereçamento suplementar ("extended addressing"), que reconhecem um endereço adicional de uma parte do instrumento, por exemplo, uma "carta". Existem ainda instrumentos com funções múltiplas, que podem ser consideradas independentes (eg., um instrumento que é composto por uma impressora e um registador ana1ógico). Neste caso estão reservados ao instrumento mais do que um endereço (tantos quantas as funções independentes que executa).

Controladores , locutores, e ouvintes

0 standard GPIB permite a designação de três classes de aparelhos no "bus". Estes são os controladores, os locutores e os ouvintes. Segue-se uma definição destas funções.

1. Controlador. O controlador é o aparelho, normalmente um microcomputador, que gere o estado do "bus" e o fluxo de informação. É ele que especifica o locutor e o(s) ouvinte(s). Sé pode existir um controlador activo de cada vez. Em configurações com mais de um controlador, apenas um *é* o controlador activo do sistema ("system controller" ou "master").
2. Locutor. Um aparelho capaz de enviar mensagens quando endereçaado pelo controlador para o fazer. Apenas pode existir um locutor de cada vez no "bus".
3. Ouvinte. Aparelho capaz de receber comandos ou dados quando endereçado pelo controlador para o fazer. Podem existir até 14 ouvintes simultaneamente.

Os controladores têm a característica óbvia de poderem funcionar, quer como locutores, quer como ouvintes, para a1ém da sua função principal de controlador. Existem ainda muitos aparelhos que implementam as duas funções de locutor e ouvinte.

Funções GPIB

Nem todos os aparelhos com a interface GPIB são capazes de responder a todos os comandos enviados para o "bus". Para a1ém das funções básicas de controlador, locutor e ouvinte, os aparelhos podem ou não ter implementadas outras funções. Assim, um aparelho com uma interface GPIB pode responder apenas a um subconjunto de todos os comandos definidos no standard GPIB. Por exemplo, um determinado aparelho pode ter implementadas as seguintes funções:

SH, AH, T5, TEO, L1, LEO, SR1, RL2, PP1, DC1, DT1, Cl

Do ponto de vista do utilizador não é fácil saber exactamente quais as funções que o instruments tem implementadas. 0 significado exacto desta lista de funções encontra-se na complexa documentação do standard IEEE-488. É, no entanto, possível tirar algumas conclusões desta lista, tendo em atenção a Tabela 4.

T	Basic Talker
TE	Extended 'I'alker
L	Listener
LE	Extended Listener
SR	Service Request
RL	Remote Local
PP	Parallel Poll
DC	Device Clear
DT	Device Trigger
C	Controller

Tabela 4: Funções implementadas por uma interface GPIB

A frente da letra (ou letras) que designam a função, encontra-se um algarismo que indica o grau de capacidade que o instrumento tem de executar a função. Zero significa que a função não está implementada nesse instrumento.

As linhas do "bus"

0 "bus" é composto por oito linhas de dados, oito linhas de controle, e oito linhas de terra. Cinco linhas de controle solo utilizadas para gerir o "bus", e as restantes três são linhas de "handshake".

Os sinais nas linhas de controle são em lógica TTL negativa. Descrevem-se, de seguida, as linhas de controle, com algum pormenor.

ATN O estado da linha de "atenção" ("attention") define se os sinais colocados nas linhas de dados devem ser interpretados como dados ou comandos. Quando a linha de "atenção" é colocada a 0 volt, os sinais nas linhas de dados devem ser interpretados pelos aparelhos como comandos; de outra forma os sinais devem ser interpretados como dados. Quando as linhas de ATN e EOI estão simultaneamente a 0 volt, está a ser feito um questionário simultâneo ("parallel poll") a todos os aparelhos.

EOI Coloca-se esta linha ("end or identify") a 0 volt para indicar o fim de uma mensagem. A maior parte dos locutores serve-se desta linha para informar o(s) ouvinte(s) do fim da mensagem, mas é também usual enviar um caracter pré-definido no fim da mensagem.

SRQ Esta linha ("service request") é colocada a 0 volt por qualquer instrumento que necessite a atenção do controlador. O controlador responde fazendo um questionário simultâneo ("paralell poll") ou sequencial ("serial poll") para determinar qual o aparelho que fez o pedido de atenção. 0 controlador deve estar programado para atender o pedido de atenção.

IFC Esta linha ("interface clear") é utilizada para inicializar as interfaces. Quando esta linha é colocada a 0 volt pelo controlador, devem ficar todas as interfaces no estado de arranque inicial e o "bus" desocupado.

REN Esta linha ("remote enable"), em combinação com a linha de ATN, coloca os aparelhos em controlo remoto.

As três linhas de "handshake" do "bus", DAV ("data valid"), NRFD ("not ready for data"), NDAC ("not data accepted"), coordenam a transferência de dados entre um locutor (um aparelho endereçado para falar ou o controlador) e os ouvintes (aparelhos endereçados para ouvir), de forma a assegurar a integridade de transferência de dados. A transferência de dados é assíncrona e a taxa de transferência ajusta-se automáticamente à velocidade do instrumento endereçado mais lento.

A sequência temporal do "handshake" está ilustrada na Figura 3.

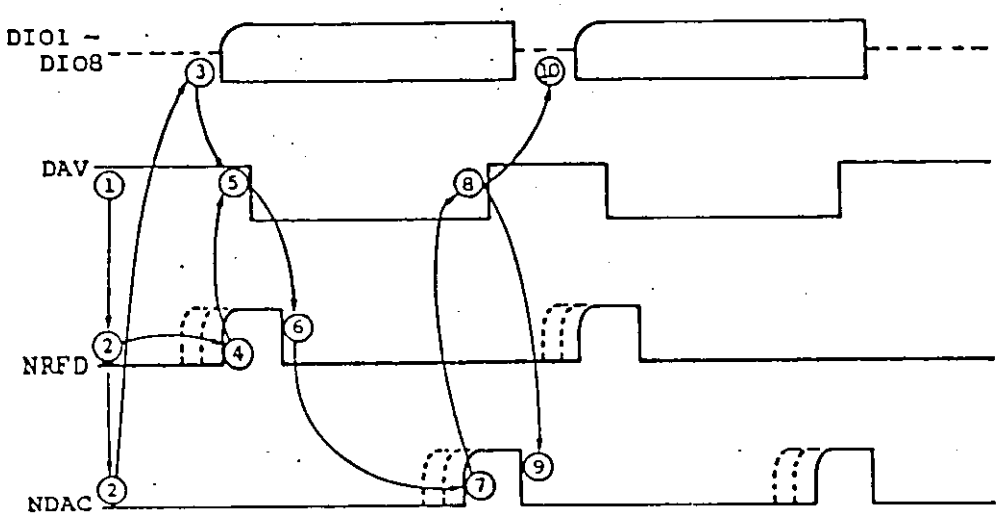

Figura 3: Sequência temporal de "handshake".

As linhas NRFD e NDAC estão ligadas em "wired or". Um locutor sé pode colocar dados nas linhas de dados quando a linha de NRFD está a 5V e a linha de NDAC está a 0V. Nessa altura coloca os dados e põe a linha de DAV a 0V.

A linha NDAC sobe para 5V apenas quando todos os ouvintes receberam o byte de dados. O locutor coloca agora a linha DAV a 5V, e em resposta os ouvintes colocam a linha NDAC a 0V e a linha NRFD a 5V, podendo em seguida iniciar-se um novo ciclo.

Os comandos do"bus'

É importante um conhecimento dos comandos do "bus" para compreender o seu funcionamento. Na eventualidade de não existirem disponíveis rotinas (um "device driver") para a interface de controlo do "bus", este pode ser controlado directamente utilizando os comandos.

Interface Message	Description	Decimal Value
PCG	*Primary Command Group*	
GTL	Go to Local	1
SDC	Selected Device Clear	4
PPC	Parallel Poll Configure	5
GET	Group Execute Trigger	8
TCT	Take Control	9
LLO	Local Lockout	17
DCL	Device Clear	20
PPU	Parallel Poll Unconfigure	21
SPE	Serial Poll Enable	24
SPD	Serial Poll Disable	25
LAG	*Listen Address Group*	
	Listen Address 0 through 30	32–62
	Unlisten	63
TAG	*Talk Address Group*	
	Talk Address 0 through 30	64–94
	Untalk	95
SCG	*Secondary Address Command Group*	
	Secondary Commands 0 through 30	96–126

Tabela 5: Tabela de comandos do "bus" GPIB

Mostra-se na Tabela 5 todos os comandos do "bus". Estes dividem-se em quatro grupos: grupo de comandos primários, grupo de endereços de ouvintes, grupo de endereços de locutores, e grupo de endereços secundários. Qualquer comando é colocado no "bus" activando a linha de ATN (pondo-a a 0V) e colocando nas linhas de dados o código correspondente. O significado dos comandos primários é na sua maioria imediato.

A maior parte dos comandos primários necessita do envio prévio de um ou mais comandos do grupo de endereços de ouvintes para seleccionar quais os instrumentos que devem responder a estes comandos. Note-se ainda que em cada transmissão de dados, apenas se pode enviar previamente um comando do grupo de endereços de locutores, ou haverá confusão no "bus".

Como já foi dito atrás, o endereço primário de cada aparelho é definido por um conjunto de cinco interruptores algures no aparelho, que permitem seleccionar um endereço entre 0 e 30 codificado em binário natural. 0 código para o comando que permite endereçar um aparelho como ouvinte é calculado da seguinte forma:

código do comando de endereço de ouvinte = 32 + endereço primário (decimal)

De igual modo, para endereçar um aparelho como locutor deve-se enviar o seguinte código:

código do comando de endereço de locutor = 64 + endereço primário (decimal)

Nenhum aparelho deve ter o endereço primáro 31, pois este endereço está reservado para de-endereçar todos os aparelhos simultaneamente:

código do comando UNListen = 32 + 31 = 63 (decimal)

código do comando UNTalk = 64 + 31 = 95 (decimal)

Todos os aparelhos obedecem aos comandos UNL e UNT.

Apenas os aparelhos com capacidade para "extended listening" (LE1) respondem a comandos do grupo de endereços secundários. Ao endereçar um aparelho com esta capacidade, através do seu endereço primário, ele espera que seja colocado no "bus" um endereço secundário. Deve ser consultada a documentação do aparelho para conhecer a sua resposta. Uma aplicação típica para este endereço secundário é seleccionar num menu uma das respostas possíveis do aparelho ao facto de estar a ser endereçado.

Subrotinas de controlo do "bus'

INITIALIZE - inicializar a interface
SEND - enviar mensagem
ENTER - receber mensagem
SPOLL - serial poll
PPOLL - paralell poll
TRANSMIT - enviar mensagem (baixo nível)
RECEIVE - receber mensagem (baixo nível)
SRQ - service request

Tabela 6: Tabela com subrotinas GPIB

Mostra-se Tabela 6 um conjunto de subrotinas típicas do software normalmente distribuído quando da aquisição de uma interface GPIB, e que variam com o computador utilizado e com o fabricante da interface (ver APÊNDICE)

Estas subrotinas não estão abrangidas pelo standard IEEE-488. São chamadas a partir de um programa escrito numa linguagem de alto nivel (eg., C). Podem existir situações em que um determinado aparelho não funciona correctamente com as subrotinas distribuidas; neste caso o programador tem que assumir o controlo directo do "bus" actuando nas linhas de controlo e enviando comandos primários.

Para a interface CEC 488 a subrotina TRANSMIT é particularmente apropriada para enviar comandos primários. Aceita (entre outros) os seguinte argumentos:

LISTEN - define um ou mais ouvintes. Exemplo LISTEN 1, LISTEN 4 9 30
TALK - define um locutor. Exemplo TALK 3
UNT - desautoriza o locutor
UNL - desautoriza todos os ouvintes
MTA - coloca o controlador como locutor
MLA - coloca o controlador como ouvinte
DATA - inicia o envio de dados (a linha de ATN é desactivada). Exemplo DATA 'R1'
END - envia o byte de fim de mensagem (line feed) e activa a linha EOI
REN - activa a linha de REN (remote enable)
EOI - activa a linha de EOI

Exemplo:

```
transmit ("REN UNL UNT MTA LISTEN 3 12 DATA ’R1’ END", &status)
```

Explicação
Todos os aparelhos são colocados em controlo remoto e desautorizados de falar e ouvir. O controlador é agora o locutor e os ouvintes são os aparelhos 3 e 12. A mensagem 'R1' é agora envia seguida do caracter de fim de mensagem e da activação da linha EOI

```
transmit ("REN UNL UNT MLA TALK 3", &status)
receive (answer, maxlength, &len, &status)
```

Explicação
Agora o controlador é o ouvinte, e o aparelho 3 é o locutor. A subrotina RECEIVE permite agora receber a mensagem que o aparelho 3 enviar.

Sintese de subrotinas

Embora não exista nenhum standard para as subrotinas de controlo do "bus" GPIB (o que provoca às vezes problemas de comunicação inesperados), a maior parte do software distribuído com as interfaces GPIB implementa estas rotinas de forma análoga.

Descrevem-se, de seguida, duas sequências de comandos primitivos que implementam as duas subrotinas mais importantes para a gestão do "bus": uma rotina para enviar dados do controlador para vários aparelhos, SEND, e uma rotina para o controlador receber dados de um aparelho, RECEIVE:

procedure SEND (device 1, device 2, ..., device n, "data")

begin

(set attention line true)

Untalk
Unlisten

My (controller) Talk Address
Listen Address I
Listen Address 2
Listen Address n

(set attention line false)

while not end of string do
Put next character on the "bus"
end-while

(Set EOI line true with last character of string)

(set attention line true)
end

procedure RECEIVE (talk address, "data")
begin
(set attention line true)

Untalk
Unlisten
Talk address
My (Controller) Listen Address

(set attention line false)

While not EOI (end or identify) do
get next data characterfrom "bus" concatenate character to string end-while

(set attention line true)

end

Note-se a semelhança entre a sequência dos argumentos na subrotina TRANSMIT da interface CEC 488 e este pseudo-código.

Bibliografia

[1] J. Bastos, Instrumentação - apontamentos manuscritos (http://diana.uceh.ualg.pt/Inst/chap07.pdf)
[2] D. Wobschall, Circuit Design for Electronic Instrumentation - Analog and Digital Devices from Sensor to Display, McGraw-Hill 1987

APÊNDICE B - Subrotinas em C para controlo da interface CEC 488

C Language Interface

This manual describes the use of C with CEC's IEEE-488 interfaces. This software has been tested with Microsoft C, version 3.0 and later and with Borland's Turbo C and C++. The same software also works with most other C compilers, as long as they support the keywords "pascal" and "far".

Interface files

The C support files are located in directory \C on the IEEE-488 applications disk. You should copy the necessary files to a working directory on your system disk. You will need:

\C\IEEE-C.H
\IEEE488.LIB

For Borland's C++, you should use:

\C\IEEE-CPP.H

instead of IEEE-C.H.

If you are writing a C program to run under the OS/2 operating system, you will need these files:

\C\IEEE-C.H
\OS2\IEEEOS2.LIB
\OS2\IEEEOS2.DLL

The IEEEOS2.LIB file should be linked with your program in place of the IEEE488.LIB file. IEEEOS2.DLL must be installed on your OS/2 system in order to run your program.

Compiling programs

First, write your C program using the IEEE-488 subroutine calls shown in the next section. Make sure to include the line:

```
#include <ieee-c.h>
```

or, for Borland C++:

```
#include <ieee-cpp.h>
```

Compile your program normally and link it with the library IEEE488.LIB.

Microsoft C

```
C> CL myprog.c /link ieee488
```

Turbo C or C++

For Borland's Turbo C, make a project file listing both your C source file and the required library. See the Turbo C manual for more information on project files. For example, create the file MYPROG.PRJ with this text:

```
myprog.c
ieee488.lib
```

Then, select this file as the current project using the Project menu. Compile and run normally (Alt-R).

IEEE-488 Subroutine calls

The following code shows the calling sequence for each IEEE-488 interface subroutine. Those arguments which are not pointers to int's or unsigned's may be passed as constants rather than variables if you wish. For example, you can call SEND as either:

```
send (addr,str,&status);
```

or

```
send (16,"this is a test",&status);
```

Note that integer constants do not contain a decimal point.

INITIALIZE

```
initialize (my_addr,level);
```

- "my_addr" (int) is the IEEE-488 address to be used by the interface card. It must be an integer from 0 to 30, and should differ from the addresses of all devices to be connected.
- "level" (int) indicates whether or not the interface card will be the system controller. It should be zero for system control mode and two for device mode.

SEND

```
send (addr,info,&status);
```

- "addr" (int) is an integer indicating the IEEE-488 address of the device to send the data to.
- "info" (char *) is a string containing the data to be sent. One or two end-of-string characters may be added to the data (see SETOUTPUTEOS later, the default is line feed).
- "status" (int *) is a variable which indicates the success or failure of the data transfer.

ENTER

```
enter (rstring,maxlen,&l,addr,&status);
```

- "rstring" (char *) is the string into which the received data will be placed. The string will automatically have a terminating null byte appended to make a valid C string.
- "maxlen" (unsigned) is the maximum number of characters desired.
- "l" (unsigned *) is a variable which will be set to the actual received length.
- "addr" (int) is the IEEE-488 address of the device to read from.
- "status" (int *) is a variable which indicates the success or failure of the data transfer.

SPOLL

```
spoll (addr,&poll,&status);
```

- "addr" (int) is an integer indicating the IEEE-488 address of the device to serial poll.
- "poll" (char *) is a variable which will be set to the poll result.
- "status" (int *) is a variable which indicates the success or failure of the data transfer.

PPOLL

```
ppoll (&poll);
```

- "poll" (char *) is a variable which will be set to the result of the parallel poll operation.

TRANSMIT

```
transmit (cmdstring,&status);
```

- "cmdstring" (char *) is a string containing a sequence of IEEE-488 commands and data. See page 3-24.
- "status" (int *) is a variable which indicates the success or failure of the data transfer.

RECEIVE

```
receive (rstring,maxlen,&l,&status);
```

- "rstring" (char *) is the string into which the received data will be placed. The string will automatically have a terminating null byte appended to make a valid C string.
- "maxlen" (unsigned) is the maximum number of characters desired.
- "l" (unsigned *) is a variable which will be set to the actual received length.
- "status" (int *) is a variable which indicates the success or failure of the data transfer.

TARRAY

```
tarray (d,count,eoi,&status);
```

- "d" (pointer to any type) is the array variable containing the data to be transmitted.
- "count" (unsigned) is the number of bytes to be transmitted.
- "eoi" (char) is zero if the EOI signal is not desired on the last byte, or one if it is desired.
- "status" (int *) is a variable which indicates the success or failure of the data transfer.

RARRAY

```
rarray (d,count,&l,&status);
```

- "d" (pointer to any type) is the array variable into which data will be received.
- "count" (unsigned) is the maximum number of bytes to be received.
- "l" (unsigned *) is a variable which will be set to the actual number of bytes received.
- "status" (int *) is a variable which indicates the success or failure of the data transfer.

SRQ

```
if (srq()) ... { put any statement here }
```

SETPORT

Note: this routine is not needed on PS488, since the I/O port address is handled automatically. It needed on the other 488 interfaces only if the I/O address is changed from the standard setting of 2B8 hex.

```
setport (board,ioport);
```

- "board" (int) is the IEEE-488 board number (from 0 to 3). Use zero if you have only one IEEE-488 board.
- "ioport" (unsigned) is the I/O port address of the IEEE-488 board.

BOARDSELECT

```
boardselect (board);
```

- "board" (int) is the IEEE-488 board number (from 0 to 3).

DMACHANNEL

```
dmachannel (ch);
```

- "ch" (int) is the DMA (direct memory access) channel to be used by the 488 interface board. If -1 is used, DMA is disabled for the current board. The default setting is -1 (disabled).

SETTIMEOUT

```
settimeout (msec);
```

- "msec" (unsigned) is the desired timeout period in milliseconds.

SETOUTPUTEOS

```
setoutputEOS (eos1,eos2);
```

- "eos1" and "eos2" (char) are the desired end-of-string characters to be appended when the SEND call is used. If only one end-of-string character is desired, set eos2 to zero.

SETINPUTEOS

```
setinputEOS (eos);
```

- "eos" (char) is the desired end-of-string character which will cause the ENTER and RECEIVE routines to terminate.

LISTENER PRESENT

```
present = listener_present (addr);
```

- "addr" (int) is the device address to be tested.
- the return value (int) indicates whether a listener was present.

GPIB BOARD PRESENT

```
if (!gpib_board_present()) ...
```

ENABLE 488EX

```
enable_488ex (e);
```

- "e" (char) is 0 to disable 488EX enhancements, 1 to enable them.

ENABLE 488SD

```
enable_488sd (e,time);
```

- "e" (char) is 0 to disable 488SD protocol, 1 to enable it. See Section Q for information on 488SD.
- "time" (int) selects the 488SD timing parameter.

Examples

Using INITIALIZE, SEND, and ENTER for a simple measurement:

```
main ()
{
     int status;
     unsigned l;
     char r[80];

     initialize (21,0);
     send (16,"F0R0X",&status);
     enter (r,79,&l,16,&status);
     printf ("Received data is '%s'\n",r);
}
```

Using TRANSMIT and RARRAY to receive binary data:

```
main ()
{
     int status;
     unsigned l;
     int d[1000];

     initialize (21,0);
     /* send a command to the device */
     transmit ("REN MTA LISTEN 3 DATA 'READ' END",&status);
     /* set device to talk and read the data */
     transmit ("MLA TALK 3",&status);
     rarray (d,2000,&l,&status);
}
```