CAN-PC Interface

CPC-PP

User Manual

EMS
THOMAS WÜNSCHE
Sonnenhang 3
D-85304 Ilmmünster
Tel +49-8441/490260
Fax +49-8441/81860

Documentation for CAN-Interface CPC-PP

Document version: V2.0
Documentation date: January 17th, 2005.

For technical assistance please contact:

EMS Dr. Thomas Wünsche
Sonnenhang 3

D-85304 Ilmmünster

Tel.:	+49-8441/490260
Fax:	+49-8441/81860
e-mail:	support@ems-wuensche.com

Our products are continuously improved. Due to this fact specifications may be changed at any time and without announcement.

WARNING: **CPC-PP hardware and software may not be used in applications where damage to life, health or private property may result from failures in or caused by these components.**

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

1 Overview

1.1 Attributes

CPC-PP offers a range of unique features which make it valuable for many CAN based applications:

- Low cost CAN-interface for IBM PC and compatibles
- Connection to the parallel printer port - usable also on notebook computers
- CiA and ISO 11898 compatible bus-interface
- Smart system with integrated microcontroller of 80C32-family
- Directly integrated into connector
- Small size for use in low-space conditions
- Modular application interface with libraries for Borland C++, Borland Pascal and Microsoft C 6.0
- Optional MS-Windows driver with DLL based API and VxD/SYS technology for high communication throughput

1.2 General description

The small size CPC-PP module provides easy access to CAN-networks using the parallel printer port of the PC. Due to the fact that no internal card slot is required, CPC-PP can also be used with laptop or notebook computers.

CPC-PP eases the development of application software on the PC. The integrated microcontroller takes load of the PC-CPU and preprocesses CAN-messages. A high level programming interface with modular design eases software development. A library of interface routines for Borland C++, Borland Pascal and Microsoft C 6.0 is included.

Power supply for CPC-PP is provided directly across the CAN-bus conforming with CiA standard DS-102. A power saving mode using variation of processor clock can be activated by software at low bus-speeds. Besides the conceptual properties also the price of CPC-PP supports low overall costs.

1.3 Sample Applications

The application area of CPC-PP is very wide. Some sample applications are detailed in the following:

- Online-configuration of CAN networks
- Network setup and analysis
- Use of PCs as CAN nodes on the application level
- Visualisation of process parameters in CAN based systems

1.4 Ordering Information

10-00-040-20	**CPC–PP** Active CAN-PC interface for printer port
10-00-140-20	**CPC–PP/EX** Active CAN-PC interface for printer port with extended supply voltage range

2 Software

The software consists of two parts which communicate across the parallel printer port of the PC. One part is executed by the microprocessor inside CPC-PP and can not be changed by the user. The application program runs on the PC and makes use of the interface library.

2.1 Functions of CPC-PP

CPC-PP offers enhanced functionality for CAN communication:

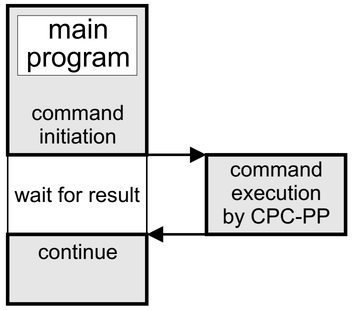
- Transmission and reception of CAN-messages
- Filtering and buffering of received messages
- Measurement of bus-load

The functions of CPC-PP are accessed across the interface library of the PC.

2.2 Application program: Realization concepts

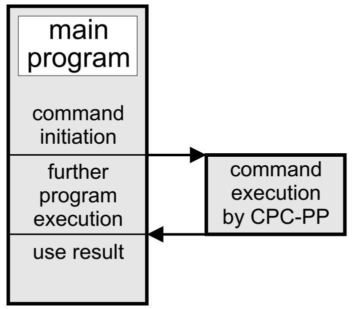
The library of interface functions supports two ways to implement the application program. The synchronous mode complies with conventional programming. The sequence of program steps is given by the program structure. Asynchronous mode allows event driven programming similar to the way used in graphical user interfaces.

2.3 Synchronous Interface



The implementation of the main program with synchronous interface allows simple and clearly arranged programs with sequential flow. It is suitable mainly for simple applications, which allow a predefinition of events to process. This is true if, for example, only CAN-messages are to be received or only bus-load measurement is to be realized.

2.4 Asynchronous Interface



The asynchronous interface provides enhanced flexibility in reaction to events which are not predictable in their sequence of occurrence. Communication objects can be processed independently of program state, reactions can be configured flexible.

For this purpose every communication object is handed on to a set of handling functions. Such functions are provided within the programming library. The application programmer can add routines as required by application purposes.

2.5 Data Structures and Library Functions

2.5.1 Data Structures

The following structures are declared in the include-file cpc.h.

Please notice that the structures and functions described in the following refer to the old DOS and Windows libraries and are included in this manual for compatibility reasons. The new structures and functions are described within the manual 'CPC Series Development Kit for MS Windows environment'.

---

struct CPC_MSG

Declaration:

```
struct CPC_MSG {
      unsigned char typ;
      unsigned char length;
      union {
            unsigned char genericmsg[];
            unsigned char textmsg[];
            char versionmsg[];
            char serialmsg[];
            struct CPC_CAN_MSG canmsg;
            unsigned char canstatemsg;
            struct CPC_CAN_PARAMS
                  can_params_msg;
      };
};
```

Description: CPC_MSG serves for parameter-transfer between application program and interface library.

---

struct CPC_CAN_MSG

Declaration:

```
struct CPC_CAN_MSG {
      unsigned short id;
      unsigned char length;
      unsigned char overrun;
      unsigned char msg[8];
};
```

Description: CPC_CAN_MSG serves for transfer of CAN-messages between application program and interface-library.

---

struct CPC_CAN_PARAMS

Declaration:

```
struct CPC_CAN_PARAMS {
      unsigned char acc_code;
      unsigned char acc_mask;
      unsigned char btr0, btr1;
      unsigned char outp_contr;
};
```

Description: CPC_CAN_PARAMS defines initialization values for the CAN controller in CPC-PP (type PCA82C200).

struct CPC_INIT_PARAMS

Declaration:

```
struct CPC_INIT_PARAMS {
      struct CPC_CAN_PARAMS
std_can_params;
      unsigned char secure_transmit;
      void interrupt (far * inthandler)();
};
```

Description: The global variable CPC_Init_Params, which has this type, holds initialization parameters.

2.5.2 Synchronous Functions

CPC_CAN_Init

Syntax:

```
#include "cpc.h"
int CPC_CAN_Init(void);
```

Description: CPC_CAN_Init() initialises the parameters of the CAN-Controller within CPC-PP. The CAN-controller is set up with parameters supplied in the global structure CPC_Init_Params (declaration in cpc.h). These parameters can be changed before the call to CPC_CAN_Init(). CPC_CAN_Init is to be called before data transmission across the CAN-bus.

Return value: -

CPC_Control

Syntax:

```
#include "cpc.h"
int CPC_Control(int);
```

Description: CPC_Control() serves for set up of the communication object types to be transmitted from CPC-PP to the PC. The upper 6 bits select the type of communication object, the lower 2 bits determine the transmission. The properties that can be influenced are described in cpc.h.

Return value: -

CPC_Exit

Syntax:

```
#include "cpc.h"
void CPC_Exit(void);
```

Description: CPC_Exit() is to be called before leaving the application program. CPC_Exit() is in any case to be used paired with CPC_Init().

Return value: -

CPC_Get_Busload

Syntax:

```
#include "cpc.h"
int CPC_Get_Busload(void);
```

Description: CPC_Get_Busload() measures the actual bus-load and returns it as percentage of the maximum bus-load.

Return value: Actual bus-load: 0 corresponds to 0%, 255 corresponds to 100% bus-load.

CPC_Get_Serial

Syntax:	#include “cpc.h”
	char * CPC_Get_Serial(void);

Description:	CPC_Get_Serial returns the serial number of the connected CPC-PP module.
Return value:	Pointer to a string with the serial number or NULL in case of errors.

CPC_Get_Version

Syntax:	#include “cpc.h”
	char * CPC_Get_Version(void);

Description:	CPC_Get_Version returns the version number of the connected CPC-PP module.
Return value:	Pointer to a string with the version number or NULL in case of errors.

CPC_Init

Syntax:	#include “cpc.h”
	int CPC_Init(void);

Description:	CPC_Init() initialises the communication with CPC-PP. CPC-PP is initialised to standard parameters, which are stored in the global structure CPC_Init_Params (declaration in cpc.h). These parameters can be changed on demand before calling CPC_Init(). CPC_Init() is to be called before usage of the other functions of the interface library.
Return value:	0 for correct initialization, -1 for initialisation errors.

CPC_Read_Msg

Syntax:	#include “cpc.h”
	void CPC_Read_Msg
	(struct CPC_CAN_MSG *);

Description:	CPC_Read_Msg() receives a message from the CAN-bus. The received communication object is stored in a structure of type CPC_CAN_MSG, which is indicated by the pointer passed on function call.
Return value:	-

CPC_Send_Msg

Syntax:
```
#include “cpc.h”
int CPC_Send_Msg
        (struct CPC_CAN_MSG *);
```

Description: CPC_Send_Msg() sends a message across the CAN-bus. The function call passes a pointer to a structure of type CPC_CAN_MSG, which contains the communication object to be transmitted.

Return value: -

CPC_Send_RTR

Syntax:
```
#include “cpc.h”
int CPC_Send_RTR
        (struct CPC_CAN_MSG *);
```

Description: CPC_Send_RTR() transmits a Remote-Transmission-Request-Message across the CAN-bus. The function call passes a pointer to a structure of type CPC_CAN_MSG, which contains the communication object to be transmitted.

Return value: -

2.5.3 Functions for the Asynchronous Programming Interface

CPC_Add_Handler

Syntax:
```
#include “cpc.h”
int CPC_Add_Handler(void (*handler)
        (const struct CPC_MSG *));
```

Description: CPC_Add_Handler() adds the handler indicated by the pointer passed at function call to the list of handlers which are executed on any incoming CPC-PP message.

Return value:	0	on error free execution,
	-1	if the list of handlers is full.

CPC_Remove_Handler

Syntax:
```
#include “cpc.h”
int CPC_Remove_Handler(void (*handler)
        (const struct CPC_MSG *));
```

Description: CPC_Remove_Handler() removes the handler indicated by the pointer passed at function call from the list of handlers which are executed on any incoming CPC-PP message. If the handler is contained more than once, the last occurence is removed.

Return value:	0	on error free execution,
	-1	if handler was not within the list.

CPC_Handle

Syntax: #include “cpc.h”
struct CPC_MSG * CPC_Handle();

Description: CPC_Handle() checks for availability of a new message from CPC-PP. If a message is available, all asynchronous handlers are called in the sequence of their entry position with the new message as parameter. CPC_Handle() returns immediately, independant of the availability of a message.

Return value: A pointer to a static memory area containing the message is returned. This memory area is overwritten during following calls to CPC_Handle() and also on use of many of the synchronous interface functions. If no message is available, CPC_Handle() returns NULL.

2.6 MS-Windows Driver Additional Information

Please notice: this subchapter refers to the older version of the Windows Development Kit. For information on the new version please read the ‘CPC Series Development Kit for MS Windows environment’ manual.

The software functionality and interface equals the MS-DOS version. Differences exist in the software setup and a few additional functions.

2.6.1 Installation

The installation is provided by the setup program. Run SETUP.EXE from delivery disk. The installation program performs the following actions:

- copy the files
- install entry for virtual device driver in SYSTEM.INI

2.6.2 Additional Functions

One additional function is necessary to compensate the fact that the initialisation structure is not directly accessible to the application:

CPC_Get_Init_Params_Ptr

Syntax: #include “cpc.h”
structure CPC_INIT_PARAMS
CPC_Get_Init_Params_Ptr(void);

Description: This function provides access to the initialization structure, which is contained in the Dynamic Link Library.

Return value: A pointer to the initialization structure in the Dynamic Link Library.

3 Electrical Characteristics

3.1 Absolute Limiting Values

Any (also temporary) stress in excess of the limiting values may cause permanent damage on CPC-PP/Eco.

Parameter	Min	Max	Unit
Storage temperature	-20	80	°C
Operating temperature*	0	60	°C
Supply voltage (standard version)	0	16	V
Supply voltage (EX version)	0	30	V
Voltage on bus connections	-4	15	V
Voltage on bus connections (EX version)	-30	30	V
Current across ground connection	–	1	A

* Extended temperature range on demand

3.2 Nominal Values

Parameter	Min	Typ	Max	Unit
Current consumption	–	40	120	mA
Supply voltage (standard version)	7	–	14	V
Supply voltage (EX version)	16	–	28	V
CAN controller clock frequency	–	16	–	MHz
Bus data rate	–	10, 20, 50, 100, 125, 250, 500, 1000 and others	–	kBit/s
Throughput to PC486/DX2-66 at CAN bitrate 125kBit/s	100	–	–	%
Throughput to PC486/DX2-66 at CAN bitrate 1MBit/s	12,5	15	–	%

4 Operating Instructions

4.1 Connection Scheme

The CAN-interface-connector (D-Sub 9 male) complies to CiA Standard DS 102-1. The pin usage is detailed in the following table.

Pin 1	–	Reserved by CiA
Pin 2	CAN_L	CAN_L bus-line (dominant low)
Pin 3	GND	Ground
Pin 4	–	Reserved by CiA
Pin 5	–	Reserved by CiA
Pin 6	(GND)	Optional ground, internally connected to Pin 3
Pin 7	CAN_H	CAN_H bus-line (dominant high)
Pin 8	–	Reserved by CiA (error-line)
Pin 9	V+CAN	Positive power supply from CAN-bus

4.2 Installation

CPC-PP operates at the parallel printer port of IBM-compatible PCs. If connected to other devices or interfaces, even if they use the same connector type, permanent damage to CPC-PP as well as to the other device or interface may result. The installation has to be done with the necessary care.

Installation may only be done with power removed from the PC as well as the CAN-bus. CPC-PP should first be connected to the PC, then to the CAN-bus. To prevent damage due to electrostatic discharge, equal electrical potential between CPC-PP and PC has to be inforced.

Power supply for CPC-PP is achieved through the CAN-bus with ground on pin 3 of the 9-pin CAN-connector, positive supply on pin 9. For proper reset it is important that supply power is switched on.

WARNING: PC-interface and CAN-bus are not galvancally decoupled within CPC-PP. The use in systems with differing ground potential between PC and CAN-bus is not allowed. Different ground potentials may also exist in systems that get ground potential from different points in an electrical installation.

Devices that are not grounded may have a floating potential. If connection across the CAN-bus between these devices is closed without external inforcement of equal ground potential, CPC-PP may be damaged permanently.