Praktikumsplatz Prozessdatentechnik OS-9 OAKEmuf

1. Beschreibung:

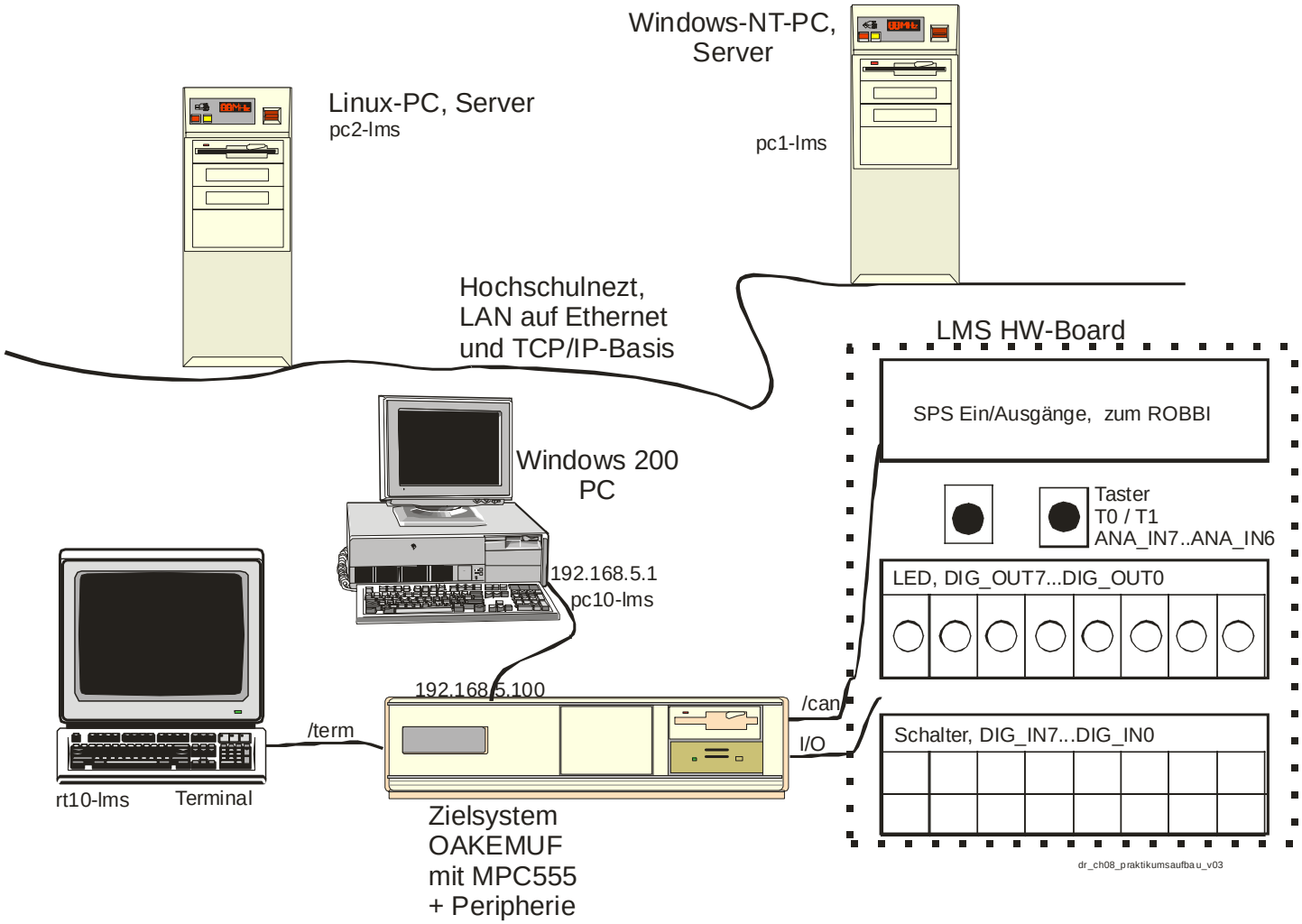

Abbildung 1,Übersicht Praktikums-Platz(PP):

Der Praktikumsplatz (PP) (siehe Bild) besteht aus einem Windows2000 **PC**, einem diskless **OS-9-System** ausgerüstet mit 4 Mbyte RAM, einem **Terminal** und dem LMS-HW-Board.

Das OS-9-System bootet nach dem Einschalten aus dem FlashRom. Im Praktikum wird OS9000 geladen, im Weiteren wird dieses System OS-9 genannt.

Das OS-9 System am Praktikumsplatz hat die IP-Adresse 192.168.5.100 und ist via Ethernet ausschließlich und direkt mit dem Windows2000 PC am Praktikumsplatz verbunden. Der Windows2000 PC besitzt zwei Netzwerkkarten:

1. eine ist mit dem OS-9 des Praktkumsplatzes verbunden und hat die IP-Adresse 192.168.5.1 und
2. die andere ist mit dem Hochschulnetz verbunden, deren IP-Adresse wird zur Boot-Zeit ermittelt.

Die OS-9-Systeme sind ‚unsichtbar' aus dem Hochschulnetz, die Windows2000 PCs routen keine Pakete vom OS-9-System ins Hochschulnetz und umgekehrt.

OS-9-System Hardware I/O-Ports:

Das OS-9-System besitzt sehr viele I/O-Schnittstellen. Die meisten davon werden direkt vom Prozessor unterstützt.

Schnittstellen Name	Descriptor	Bemerkung
Serielle Schnittstelle 1	/term	Verbunden mit dem Terminal
Serielle Schnittstelle 2	/t2	
Digitale Ausgänge über TPU_A		ATPUCH7…ATPUCH0
Digitale Eingänge über MIOS		MPIO32B13…MPIO32B6
Analoge Eingänge		AAN51…AAN0
CAN		Can-Bus
Ethernet	/spsmc0; /enet	

An der seriellen Schnittstelle des OS-9-Systems ist das **Terminal** angeschlossen. Die Übertragungsrate beträgt 9600 Baud.
An der parallelen Schnittstelle des Echtzeitrechners ist das **LMS-HW-Board** angeschlossen. Die parallelen Schnittstellenfunktionen sind im PowerPC 555 Chip onboard integriert.
Es gibt acht Digitale Ausgänge [(CPU-MSB: ATPUCH7/ PIN: DIG_OUT7/ ST3: Pin 12)-(ATPUCH7/DIG_OUT7/Pin 5)] an denen die acht LEDs angeschlossen sind.
Es gibt acht digitale Eingänge [(CPU-MSB: MPIO32B13/PIN:DIG_IN7/ST4: Pin 9)- (MPIO32B6/DIG_IN0/Pin 2)] an denen die acht Schalter angeschlossen sind.
Es gibt acht analoge Eingänge :
[(CPU-MSB: AAN51/PIN: ANA_IN7/ST4: Pin 10)- (CPU-MSB:AAN48/PIN: ANA_IN4/ST4: Pin 13)…….
(CPU-MSB: AAN3/PIN: ANA_IN3/ST4: Pin 14) - (CPU-MSB:AAN0/PIN: ANA_IN0/ST4: Pin 17)]
Der Taster T0 ist an ANA_IN7 und der Taster T1 an ANA_IN6 angeschlossen.
Es gibt einen CAN-Bus auf dem Echtzeitrechner der am Praktikumsplatz an die mittels CAN-Bus versehene SPS-E/A-Gruppe angeschlossen ist.

OS-9-Bedienung und Allgemeines:

- Die lokale RAM-Disk hat die Laufwerksbezeichnung **/dd (default-disk). Nur auf diese RAM-Disk können Sie schreiben!!.** Jedes **OS-9-System** ist Fest-/Wechselplattenlos.
  Vorsicht: die RAMdisk verliert durch das Ausschalten oder Reboot des OS-9-Systems ihren Inhalt!!
- **Terminalbedienung:** Beim Booten des Systems erscheinen am Terminal die einzelnen Bootmeldungen. Das Booten des OS-9 Systems ist nach c.a. 5-10 Sekunden abgeschlossen und es erscheint der mshell $ Eingabeprompt. Ab jetzt ist das System einsatzbereit und Sie können interaktive Kommandos geben. Geben Sie logout ein erscheint sofort wieder der mshell Prompt $. Damit bei Kommandos der Bildlauf nicht seitenweise erfolgt, sondern kontinuierlich, geben sie am Prompt das Kommando: ***tmode nopause*** ein.
- **Weitere interaktive Terminals via Telnet-Verbindung vom Windows2000 PC aus:** Um interaktiv vom PC aus mit dem OS-9 System zu arbeiten, können Sie eine **telnet** Verbindung zum OS-9-System öffnen. Kommando am PC (➔Ausführen wenn kein Telnet-Icon vorhanden ist) : **telnet 192.168.5.100**
  Danach erscheint die Abfrage:
  ➔ OS9-Username: super Passwort: user
- **Die dann laufende Shell besitzt eine Startpriorität von 128.** Prozesse mit dieser Priorität werden im Timesharing bearbeitet.

2. Manuals zur C-Bibliothek/OS9 nur online

Siehe: http://cd1.ee.fhm.edu/Praktikum_Prozessdatentechnik/os9_hilfe.html

3. Windows 2000-PC-Systeme (Anmeldung und Verzeichnisse)

- **Anmeldung PDT Praktikum:**
  PC-Username: PDT01 – PDT16 Passwort: pdt
- **Anmeldung Echtzeitsysteme-Praktikum:**
  PC-Username: OS9<Platznummer> Passwort: os9
- **PC-Arbeitsverzeichnisse für beide Praktika**: **L:\ Projekte**

4. Windows 2000 PC-Systeme als OS-9-Cross-Entwicklungsplätze

- Sie erstellen am Windows2000 PC ihre Programme mittels der integrierten Entwicklungsumgebung „**Microware Hawk IDE“**
- **(hawk) Kurzanleitung:**
  - hawk integriert Editor/Compiler/RemoteDebugger für das OS-9-System.
  - **hawk strukturiert den Entwicklungvorgang in folgende Ebenen:**
    - 1. oberste Ebene: **Projectspace** entspricht einer Sammelmappe für **ein oder mehrere Projekte**
      ➔ In unserem Fall entweder: Projektspace: L:\Projekte\PDTPraktikum oder L:\Projekte\OS9Praktikum
    - 2. Ebene: **Projekt**, ein Projekt ist ein in sich abgeschlossenes Ganzes, das aus einer oder mehreren **Components** besteht.
      ➔In unserem Fall: **Projekt: Versuch1, Versuch2 ...Versuch3...**
    - 3. Ebene: **Component, der Name einer Component ist bei der Erstellung von Programmen gleichzeitig der Name der ausführbaren OS9-Programmmodul- Datei** , eine Component enthält alle Source- und Header-**Dateien (Units)** zu einem lauffähigen Programm
    - 4. Ebene: **Units :** einzelne **Dateien entweder .c oder .h**
  - **hawk merkt sich i.d.R. den letzt geöffenten Projektspace und öffnet diesen bei Wiederstart**

- **Erster Praktikumstag:**
  - **Starten von hawk und Anlegen eines Projectspaces ➔ Menu: projects/projectspace/new**
    - **Current Directory L:\Projekte, Filename entweder PDTPraktikum oder OS9Praktikum, Die Aufforderung Projekte hinzuzufügen verneinen, Warnung ignorieren**
    - **Damit wurde für das gesamte Praktikum ein Projektspace angelegt**

- **Bei jedem neuen Praktikums-Versuch: Anlegen eines neuen Projektes ➔ Versuch1 ....Versuch3**
  - **Menu: projects/projectspace/Add New Project**
  - **Projectname: Versuch<x>, Folder: L:\Projekte**, **Chip M505**
  - **Component: Name des zu erstellenden ausführbaren OS9-Programmmoduls z.B. hallo oder systask, nächstes Menü Units mit finish beenden**

- **Anlegen einer Component innerhalb eines bestehenden, neuen Projects:**

  - **Project auswählen im Projectfenster und <rechte>-Maustaste drücken und 'new-component' wählen**
  - **Component: Name des zu erstellenden ausführbaren OS9-Programmmoduls z.B. hallo oder systask, Chip M505 wählen, nächstes Menü Units mit finish beenden**

- **In jedem (Teil-)Versuch für jede neue .c oder .h Datei (Unit):**

  - In der Toolbar 'create new empty document' auswählen, sofort Menu: File/Save as/ auswählen und **<filmanen>.c oder .h eingeben, dabei achten ob die Datei auch in L:\Projekte oder einem Unterverzeichnis davon angelegt wird.**

  - **Wichtig: Danach im Editorfenster <rechte>-Maustaste drücken und 'add File to Project' auswählen ➔ 'Into Component': die Component auswählen in die die Datei aufgenommen werden soll und mit <ok> den Vorgang abschliessen.** Jetzt gehört die Datei zur Component dazu.

- **hawk-Project-Ansicht:**
  - Fenster links oben: Components-Baum ➔ Ansicht Wurzel entspricht Projektebene Versuch1,Versuch2...
    Jeder **Praktikumsteilversuch** ist je als eine neue Componente anzulegen
  - Fenster links unten: zeigt alle Components wenn im Fenster links oben die ‚Wurzel' angewählt ist, ansonsten die zu einer Component gehörigen Dateien
  - Fenster rechts oben: Editor Fenster
  - Fenster rechts unten: Fehler Fenster für Compiler
- **Source-Dateien (Programme) übersetzten/laden/debuggen:**
  - **Vor jedem Übersetzten <save-all> im Filemenü auswählen.** Übersetzen:➔ Componente anwählen, rechte-Maustaste, <build>/<rebuild> auswählen ➔ Fehlerausgabe im Build-Fenster beachten.
  - Soll das fertige fehlerfrei übersetzte Programm ausprobiert werden, gibt es zwei Möglichkeiten: Laden/und am Terminal manuell starten **oder** vom PC aus debuggen.
    - Debuggen ➔ Componente anwählen, rechte-Maustaste, <debug> auswählen, beim ersten Mal die Targetfrage mit 192.168.5.100 beantworten (➔Bedienung Debugger intuitiv).
    - Laden und manuell starten: ➔ Componente anwählen, rechte-Maustaste, <load> auswählen, beim ersten Mal die Targetfrage mit 192.168.5.100 beantworten; dann am Terminal/Telnetverbindung den Programmnamen eintippen und das Programm startet ;-)).
    - **Achtung : Nach jedem fehlerfreien Neuübersetzen und vor dem Laden/Debuggen mittels procs sich vergewissern ob noch das Programm läuft. Wenn ja dann mit kill <taskid> beenden. Danach mittels unlink <Programmname> sicherstellen, dass das Programmmodul auch aus dem Moduldirectory entfernt wurde.**
- **PC-FTP** dient zum Übertragen ihrer am PC erstellten Dateien an Ihr OS-9-System. Nach dem Starten des Programms wird mittels **connect** Verbindung aufgebaut und eine bestehende Verbindung mittels **close** abgebaut. Ein **FTP** -connect funktioniert nur wenn das OS-9-System gebootet ist und nicht ´hängt´ (➔Taskprioritäten beachten!!). Unter den genannten positiven Umständen braucht die Verbindung für mehrere Filetransfers nur einmal hergestellt werden.
  ➔ **connect-**Aufruf: Suchen sie in der Listbox die Verbindung zu Ihrem OS-9-System ➔falls nicht vorhanden
  Aufsicht fragen ➔ starten der Verbindung und das PC- FTP-Programm loggt sich auf Ihrem OS-9-System ein und ´spricht´mit dem sog. FTP-Dämonen (Server) Ihres OS-9-Systems➔ es erscheinen zwei Boxen ➔links das PC-Filesystem und rechts das OS-9-Filesystem an der root von /dd ➔ Stellen Sie links ihr Arbeitsdirectory ein ➔ es erscheinen ihre Dateien, markieren Sie nun die zu übertragenden Dateien ➔ mittels => -Button übertragen Sie nun von PC nach OS-9-System die markierten Dateien. ➔ Kontrollieren Sie mittels dir-Kommando am Terminal.

5. Beschreibung der Software zur Anbindung des LMS-HW-Boards an eine Anwendungstask

Es gibt zwei Varianten zur softwaremäßigen-Anbindung der parallelen Schnittstellen des LMS-HW-Boards (ohne CAN-Bus). Die Auswahl der Variante für Ihren Versuch entnehmen Sie Ihren aktuellen Praktikumsversuchsanleitungen Ihrer Lehrveranstaltung.

Variante A

- Zur Systemstartzeit (oder manuell) werden automatisch zwei system-state-Tasks gestartet: **txirqsim und lmsdigio**. Diese Tasks dürfen, da sie system-state Tasks sind, auf die Hardware direkt zugreifen und bedienen exklusiv die oben unter 1. beschriebenen Hardwareschnittstellen. Beide Tasks stellen entsprechende Software-Schnittstellen im OS-9-System jedem anderen **Anwenderprogramm, das im user-mode ausgeführt wird,** so zur Verfügung, dass alle Schalter- und Tasterstellungen eingelesen und die LEDs angesteuert werden können.

- **Variante A** erkennen Sie daran, dass mittels **procs –e** -am Terminal eingegeben- beide Tasks **txirqsim und lmsdigio** in der Taskliste erscheinen. Sollte ein manueller Start der Programme am Terminal erforderlich sein, dann starten Sie die Programme vom Terminal aus via: **txirqsim <>>>/nil& ; lmsdigio <>>>/nil&** . Einmal gestartet lassen sich diese beiden Programme **nicht** durch **kill** wieder entfernen; nur noch durch **erneutes booten**.

- **txirqsim:** txirqsim legt zur Systemstartzeit zwei Events an, die von jedem gelesen werden können: **LMS_H1 für Taste T0** und **LMS_H2 für Taste T1**. Die Events sind mit sinc=1 und winc=0 angelegt. Mit jedem Tastendruck T0/T1 wird das entsprechende Event um eins hochgezählt. txirqsim tastet die Tasterstellung alle 33 ms ab.

- **lmsdigio:** lmsdigio legt zur Systemstartzeit das Event **LMS_DOIO** und das Datenmodul **lmsdio** an; beides kann jeder lesen und schreiben. Das Event ist mit sinc=1,winc=0 angelegt. Das Datenmodul **lmsdio** ist wie folgt aufgebaut:

```
#ifndef _LMSDIO
#define _LMSDIO

#define LMSDOIO "LMS_DOIO"
#define LMSDATMOD "lmsdio"

typedef struct _lmsdio {
        unsigned char digiout;
        unsigned char digiin;
} lmsdio;

#endif
```

  - **lmsdigio Funktion:** lmsdigio wartet darauf, dass das Event LMS_DOIO auf eins (1)-gesetzt wird. Danach wird der Wert **digiout** des Datenmoduls an die LMS-HW-Board LEDs ausgegeben und sofort die LMS-HW-Board Schalterstellung eingelesen und nach **digiin** in das Datenmodul gespeichert. Danach setzt lmsdigio das Event LMS_DOIO auf den Wert Null (0)-zurück und wartet wieder auf den Eventwert LMS_DOIO = Eins.
  - **Nutzung von lmsdigio für Ein-/Ausgabe durch eine selbstgeschriebene Anwendertask:** Die Anwendertask linkt sich auf das Datenmodul und das Event von lmsdigio, schreibt einen Wert nach **digiout** in das Datenmodul **lmsdio** und setzt danach das Event LMS_DOIO auf eins. Anschliesssend wartet die Anwendertask bis das Event wieder den Wert Null hat und liest **danach!!** den Wert **digiin** aus dem Datenmodul **lmsdio** als neue Schalterstellung ein.

Variante B

- Es existiert ein Treiber **lmsboard** und der dazugehörige Descriptor **dlmsboard**. **Das für den Anwenderprogrammierer (Praktikumsteilnehmer) nutzbare OS9-Gerät heisst: /dlmsboard** . Der dazugehörige **Treiber lmsboard** darf auf die Hardware direkt zugreifen und bedient exklusiv die oben unter 1. beschriebenen Hardwareschnittstellen. **Erst durch** das Kommando **iniz dlmsboard** wird **die Hardware und die Zugriffe darauf durch den Treiber lmsboard initialisiert** (Auch die Events existieren erst ab dem Zeitpunkt).

- **Das Gerät /dlmsboard** wird vom sog. SCF-Filemanager verwaltet und aufgerufen. Der SCF-Filemanager verwaltet auch das angeschlossene Gerät Terminal /term an dem Sie interaktiv arbeiten. Das **Gerät /dlmsboard** arbeitet demnach **wie ein seriell angeschlossenes Gerät** auf das man **byteweise schreiben** oder von dem man **byteweise lesen** kann.

- **Öffnen/Schliessen des Gerätezugangs auf /dlmsboard: Bevor** die Schalterstellung gelesen und die LED-Belegung geschrieben werden kann, muss der Programmierer, wie bei jedem Datei-/Gerätezugriff, das Gerät mittels ➔ ***_os_open(...,S_IREAD | S_IWRITE,...)*** öffnen und nach Gebrauch mittels ➔ ***_os_close(...)*** wieder schliessen. Das Gerät ist **jederzeit** bereit **ein** Byte zu lesen und **ein** Byte zu schreiben. *(#include modes.h)*

- **Lesen der Schalterstellung: Mittels *_os_read(...)*** mit Lesen **eines Bytes** wird die aktuelle Schalterstellung eingelesen.

- **Schreiben der LED-Belegung: Mittels *_os_write(...)*** mit Schreiben **eines Bytes** wird der Wert an die LEDs ausgegeben.

- **lmsboard:** lmsboard legt **nach Initialisieren mittels iniz dlmsboard *zwei*** Events an, die von jedem gelesen werden können: **LMS_H1 für Taste T0** und **LMS_H2 für Taste T1**. Die Events sind mit sinc=1 und winc=0 angelegt. Mit jedem Tastendruck T0/T1 wird durch die Interuptserviceroutine des Treibers lmsboard das entsprechende Event um eins hochgezählt.

- Soll das Gerät /dlmsboard wieder **endgültig geschlossen und aus dem System, entfernt werden**, so geht das nur wenn kein Programm dies Gerät gerade benutzt, mittels des Kommandos **deiniz /dlmsboard**.

- Mittels des Kommandos **irqs** und **devs** kann man sich einen Überblick über **initialisierte Geräte** und **Treiber** und deren Resourcenbelegung verschaffen.

6.Blockschaltbild OAKEMUF:

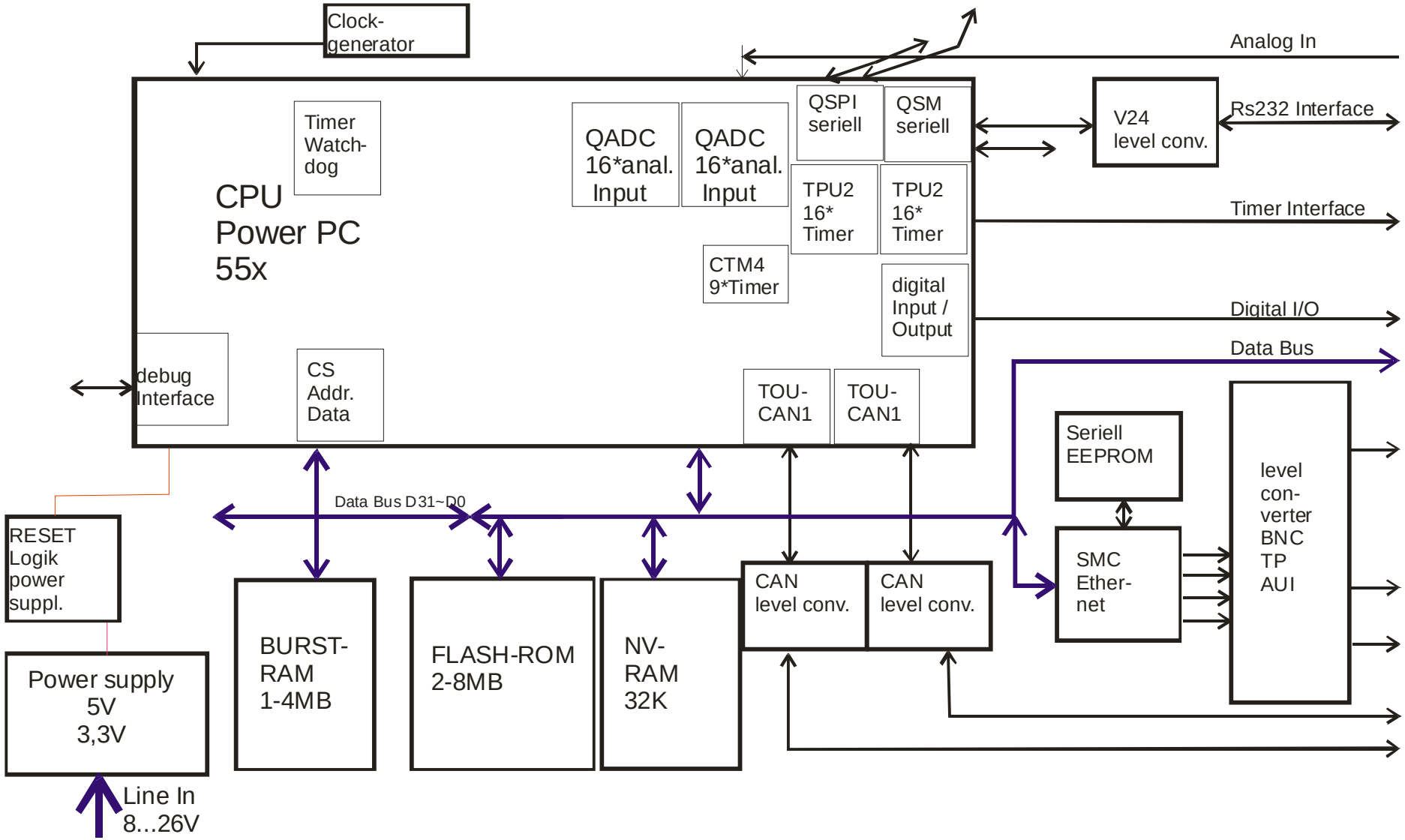

Abbildung 2, Blockschaltbild OAKEMF