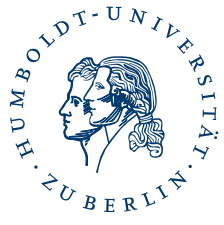


Drift Chamber Control and Readout System

Technical Description

Experimental Particle Physics
Department of Physics
Humboldt University Berlin

Olf Epler - August 2006

Contents

Chapter 1

Introduction

This documentation is focused on technical aspects - hardware and software - of a drift chamber developed and build up in cooperation between DESY-Zeuthen and the physics department of the Humboldt University Berlin.
First calibrations were done in 2005 with a meanwhile changed release and publicated in [01].
It is now possible to read out all 72 drift tube channels because the formerly used four 16 channel LeCroy TDCs 1176 are replaced against one 128 channel TDC CAEN V767.
Furthermore an adapted VME trigger system is now in use.
Important software parts are rewritten and additional features integrated.
Details are documented in the DRC Users Manual attached to this documentation.

Figure 1.1: Drift Chamber

Chapter 2

Drift Chamber Hardware

2.1 Gas System

All 72 drift tubes are coupled in series with a gas bottle equipped with gas flow and pressure regulators. The chamber gas is a mixture of 80% Ar [gain part] and 20% CO_2 [quench part]. The gas pressure must be high enough to compensate the small gas leakage and is adjustable with a flow meter regulator. The indication of a proper gas pressure and consumption is a bubble frequency around 1Hz at the oil filled gas flow controller. A gas pressure of around 1bar corresponds to approximatly 25 - 30 units at the flow meter scale.
It is important to fill the drift tubes at least one hour before the anode high voltage is turned on!
If the chamber was not used for longer time the gas system should switched on around 2 - 3 hours before the measurment starts.

Figure 2.1: Gas Regulator and Flow Meter

Figure 2.2: Gas Flow Controller

2.2 Drift Tubes and Frontend Electronics

The drift tubes and also the frontend electronics were designed for high energy experiments at CERN (ATLAS) and DESY (HERA-B). Details are publicated in several papers - only a selection is [02],[03],[04].
Each of the three drift chamber sections includes 24 drift tubes - mechanical mounted in three layers with 8 tubes per layer in a honeycomb construction.
The exact geometrical dimensions are depicted in fig. A.1 "Channel Map and Geometry".
Electrical each section is conducted to a HV plate shown below.

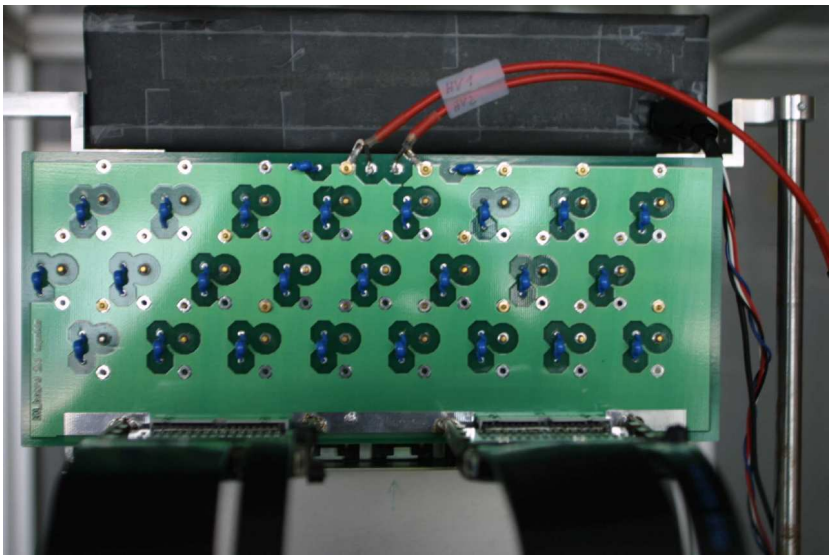


Figure 2.3: HV plate

The HV plates are the "distributors" for the positive voltage of around 2200 - 2400V. Each inner thin tube wire - the anode - is connected to this plate. The tube wall is grounded - so that an electrical field as function of the radial distance from the anode wire is build up between the anode surface a and the inner surface of the wall b after the high voltage V_0 is turned on.

$$E = \frac{1}{r}\frac{V_0}{ln(b/a)} \tag{2.1}$$

Decoupled over a capacity elementary charges - deponized by the drift electrons on the anode surface - can now as a signal amplified, shaped, and discriminated with the frontend module ASD-8. Each ASD-8 module includes two 8 channel chips - in summary 16 channels per board.
Depicted in fig. A.1 "Channel Map and Geometry" the left 12 anodes and the rigth 12 are coupled with one board respectivly.

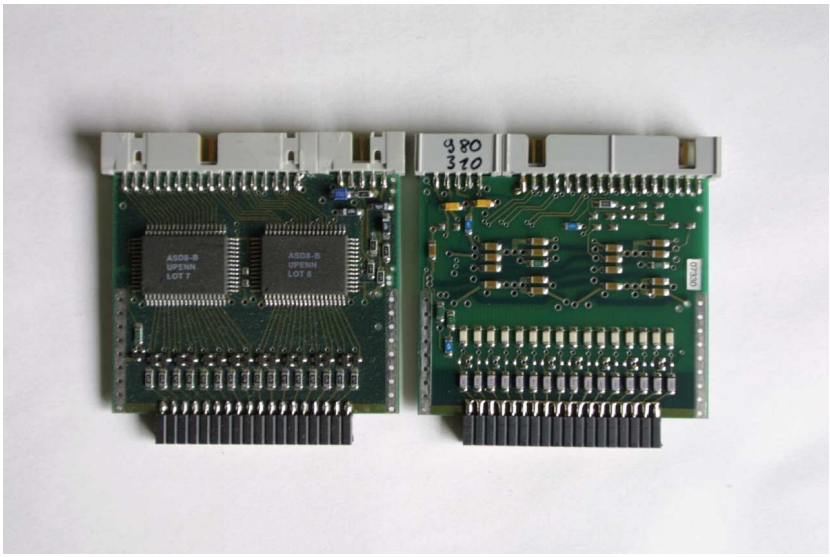


Figure 2.4: ASD-8 boards

The ASD-8 outputs are connected over shielded flat signal cables with level converter modules - ASD-8-ECL-Translator (A.3) - placed in a VME crate. The level converter outputs are standard ECL (emitter coupled logic) signals and directly connected with TDC input channels.
Since the hardware upgrade end of 2005 a CAEN TDC V767 (A.3) is now in use. The TDC time resolution is 0.8ns (LSB equivalence). Because only a common stop **emulation** run mode is provided, the delayed trigger (common stop) line is additional connected as reference channel to a TDC input. To allow a delay fine calibration in this case a second reference channels is also connected. The sum of both delays (GDU output channel 0x03) is the total hit conversion time window.
To minimize the signal noise it is necessary to set appropriate thresholds for each ASD-8 module. This is now possible separatly with fine tune potentiometers for all 6 ASD-8 boards. The potentiometers are avalaible at the front cover of the distribution board DIB (A.3). This board contains also a central power supply for all ASD-8 modules. The positive and negative power voltages are adjustable with front cover mounted fine tune potentiometers.
The power cables are small shielded flat cables - separated from the signal cables.

2.3 Trigger System

The trigger and readout electronics is placed in a VME (versa module euro card) crate and controlled by a single board computer HP 9000 743 in the left slot of the crate as shown below.

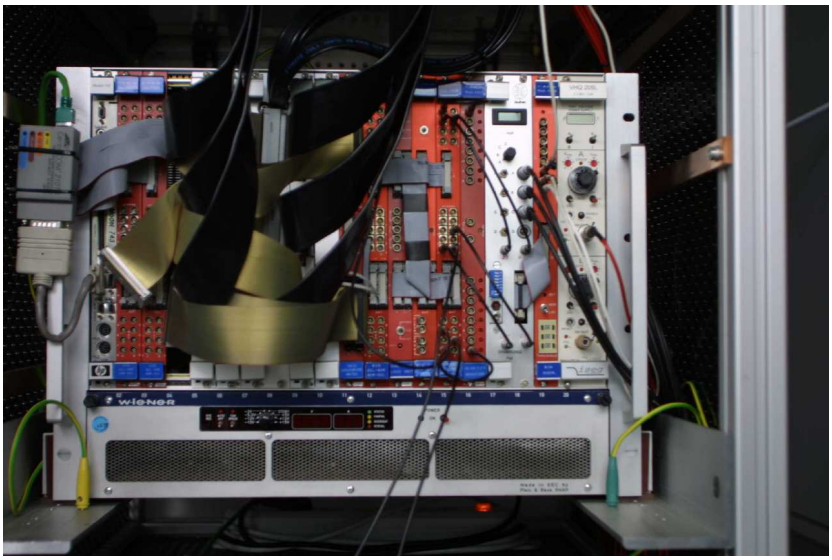


Figure 2.5: VME based DRC electronics

The VME modules and also the single board computer are connecteded to the crates data backplane. The VME CPU processes a UNIX system - HP-RT - with real time kernel extensions and a number of special VME read and write functions.

Each time a myon (μ^+ or μ^- as result of a particle decay in the higher earth athmosphere -> pion decay) crosses the drift chamber scintillator blocks an initial trigger is generated. Γ-quants - emitted by the build in organic scintillator plates in UV frequency range and transformed by wavelength shifters - pass the internal coupled ligth fibers and hit the cathode of a PMT (photo multiplier tube) Hamamatsu H5783.
Based on the light electrical effect the PMT cathode emits electrons. Accelerated by an electrical field between the cathode, a number of dynodes and the anode - additional amplified by an avalanche process - the resulting electron flow is as a short signal puls at the PMT output available.

The three scintillator blocks are mounted above the top drift tube section and below the lower two sections.
The PMT output signals - discriminated and width shaped with a discriminator DIS CAEN V258 (A.3) - feed now the inputs of the coincidence unit COC (A.3). With DIP switches - available at the front cover of this module - the input channels of the COC circuit can be activated (right position) or deactivated. The

resulting width shaped coincidence output signal COO is the trigger input for the following shown circuit.

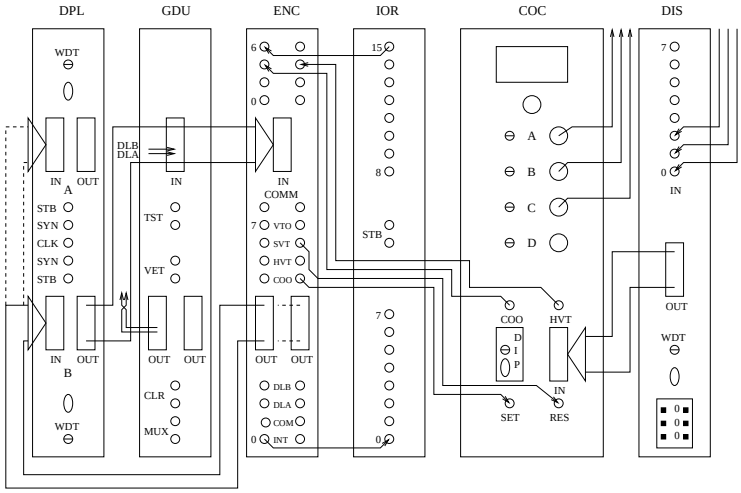


Figure 2.6: VME Trigger Circuit

The high voltage for all PMTs is derived from a control voltage and adjustable with potentiometers in the range from 0 to 1 V. Based on the coincidence output signal COO, the state of a software controlled veto SVT and the state of a hardware switched veto HVT all other signal lines are set as depicted below.

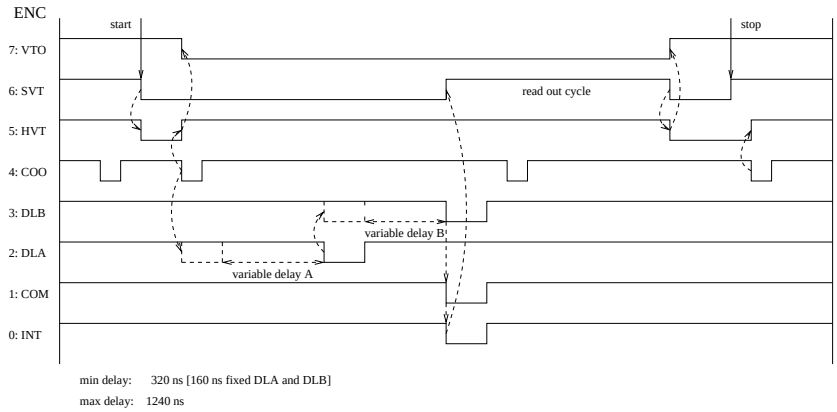


Figure 2.7: VME Trigger Diagram

Core of the trigger circuit is a dual programmable logic unit DPL CAEN V495 (A.3). During the trigger initialization the internal lookup table contents of the DPL must be programmed. Furthermore the variable delays of the gate and delay unit GDU must be set. The resulting total delay is the time window during hits - the TDC channel input pulses - can be converted into digital time values.
The total delay is adjustable between 320ns and 1240ns and set to 1000ns.
This value has to be set also as equivalent time window and offset parameter in the TDC configuration.

Additional the GDU output lines DLA (channel 0x02) and DLB (channel 0x03) are connected with two TDC reference input channels (0x0F and 0x1F).
A total delay DLT fine adjustment is therefore in following sequence possible:
1.) compute DLB = T(ch 0x1F) - T(ch 0x0F) - noise measurment
2.) set DLB so that DLT = 2 * DLB
3.) set DLA = DLB
DLA and DLB must be set as equivalent parameters in the GDU configuration. The programmable GDU multiplexer output can be used to check the settings with an oscilloscope.

2.4 TDC Readout System

The TDC readout is interrupt controlled by a VME single board computer and attached to interrupt level 1. A standard VME system has 6 levels.
Additional at each level it is possible to select different interrupt capable modules by an one byte wide interrupt vector.
The used interrupt generator module is an input output register IOR CAEN V513 (A.3). All 16 channels of the IOR can be set as input or output. In this case channel 0 is programmed as input - all other channels are outputs. As depicted in figure 2.6 "VME Trigger Circuit" channel 0 is connected to the interrupt line INT. Simultaneous to the TDC conversion start, the IOR generates an interrupt. During the following read out cycle the interrupt is deactivated, the software controlled veto line SVT is activated and the software signal SIGIO is send to a suspended readout handler process.
All these command calls are processed inside a device driver module. If the handler has received this signal, the process can now continue and leave the sigwait state. Following all run queue modules can now read out and the data written into a file or IO stream. Finally the veto line SVT is deactivated and the handler suspends again.
If a veto line is active, no further interrupts can be generated. To avoid interrupts after the detection of the first initial trigger, a further hardware switched veto line HVT is set by a flip flop circuit as long as the veto line SVT is deactived again. This circuit is also placed in the COC module together with the coincidence circuit and the PMT power supplys.

Chapter 3

Drift Chamber Software

3.1 Overview

As shown in the figure below the control and readout software includes a part processed on the real time operation system HP-RT and another part on a LINUX system. Both parts are written in one program text segment so that mostly all functions are available for each side. Operation system specific parts are separated by compiler directives or specific loaded from libraries.

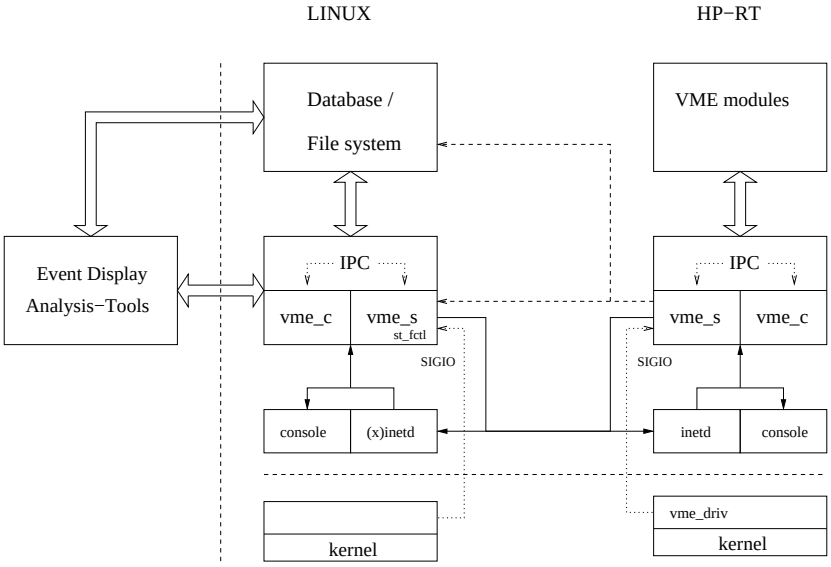


Figure 3.1: Drift Chamber Readout and Control System

Two libraries are linked with the main program.
Both libraries - odb_lib -> libiodb.a/libiodb.so and vme_lib -> libivme.a/libivme.so - provide shared used and operation system specific parts.
The main program vme_s can process different modes - client from a terminal console, client in server mode, command server and readout or data handler

server. The several server modes are invoked by the (x)inetd super server daemon depend on call parameters set in the (x)inetd configuration for different ports.
A symbolic link to the main program allows to start the client with different names from different locations inside the file system.
The server processes perform a connection check for calling hosts which must explicitly be listed as allowed hosts - except localhost. Furthermore the daemon server processes change the group and user identification equal to the clients ID. So a readout queue run state query is allowed but it is not possible to stop this queue if the UID and GID differs.
Additional to the program parameters there are a number of global parameters available. Only the global parameters read in by the client are valid for all peer servers. A detailed description of all available global and program parameters is given in the drift chambers user manual (C).
In some cases it is possible and necessary to overwrite global with program parameters. Optional the default global parameter file can be replaced by a user defined file or database table.
To decouple the main program from specific access functionalities the library odb_lib defines functions to allow the exchangeable use of sockets, files and database tables. Underlying parts of the inter process communication [IPC] between the data handler service on LINUX side and a state and event log interface is also defined here.
Furthermore a line parser and a line writer allows to read and write configuration and parameter strings from or into streams, files or database tables.
The second library - vme_lib - provides the initialization and readout functions for the VME modules and parts of the IPC system.
A more detailed library description follows in chapter 3.4 and 3.5.

3.2 HP-RT device driver

To provide the asynchronous interrupt service a device driver is linked with the HP-RT kernel. In summary 8 different handler processes can run in parallel. Therefore each readout handler has an own interrupt service. During the readout process initialization the appropriate driver is initialized. If the interrupt level between driver and VME module configuration matches, the interrupt vector is delivered to the driver -> vme_init_int [vme_lib]. Later the handlers process ID, the VME interrupt module type, the modules VME address, the address modifier byte and a veto line SVT corresponding output value is delivered by several ioctl calls -> vme_set_drv [vme_lib]. If the interrupt line of the VME system becomes active the kernel breaks and continues with an interrupt acknowledge cycle. Underlying kernel functions read out the interrupt level and vector of the VME interrupt generator module. If level and vector matches, the drivers internal interrupt service routine [ISR] is processed. The SVT veto line is set, the VME modules interrupt registers are reset, the interrupt line is cleared and the ISR sends to the handlers PID the signal SIGIO -> vme_dev_isr [vme_driv]. The handler process is suspended (sigwait state) until this time.

Now this process can continue and read out all run queue attached modules. If this is completed the handler sets the interrupt registers again and deactivates the SVT veto line -> vme_set_vto [vme_lib]. It's now possible to repeat the interrupt cycle at the next valid trigger.

3.3 Main Program

Core of the main program - independent from the run mode - is a finite state machine [FSM] processed as followed:

sta(n)	:=	F(cmd(n), f(sta(n - m)))	m ϵ 1..N	sta:	FSM state
out(n)	:=	F(sta(n - m))	m ϵ 0..N	out:	FSM output
cmd(n+1)	:=	F(out(n), sta(n))		cmd:	FSM input

To each new state can attached an output function. The return value is the following input command. The FSM loop ends at failure or if the new state is equal to the initialization state. To transfer data between attached processes, an inter process communication system [IPC] is integrated. It is controlled by the signals SIGUSR1, SIGUSR2, SIGTERM and SIGIO. The general principle is shown below:

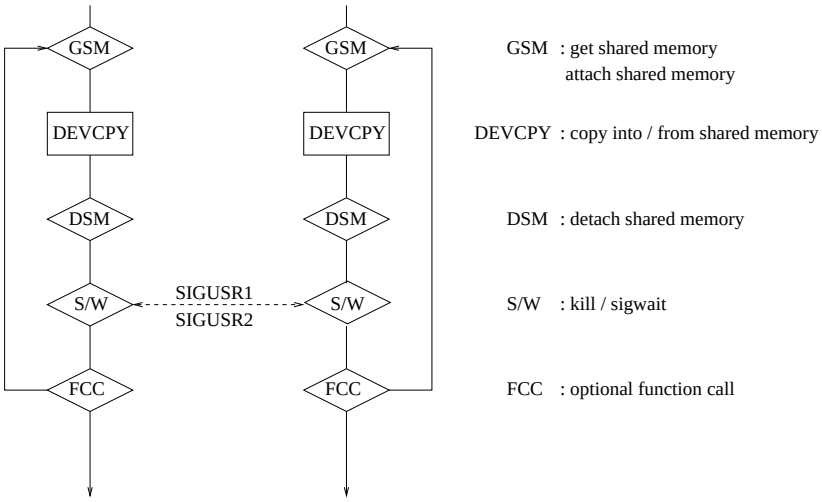


Figure 3.2: IPC sheme

To transfer data between process A and process B a shared memory device is used. This must requested before and released later - it depends on the specific program flow. The initial process - defined as client - attaches this shared memory device and copies data from the process memory into the shared memory. The peer process - defined as server - is responsible to initialize the shared memory at least with a process identification number (PID) - typical the servers own PID. If the client needs a server response, then also the clients PID must be copied into the shared memory. Because both processes can act as client or as server a client and a server receiver signal should be defined. If the client or the server has finished the copy procedure the initial process can now send a signal to the peer. The suspended peer process can now continue and attach the shared memory, copy the data into its own process memory and finally detach the shared memory.

The different IPC schemes are depicted in appendix B.2. The actual run state

and other queue informations exist as a copy in each queue owned shared memory. Handler and command servers are invoked from the (x)inetd super server daemon. Each handler process is responsible for requesting and freeing the shared memory resources.

To allow a flexible way to read and write configuration data and state informations from or into sockets, files or database tables the library odb_lib provides this functions. This allows to handle sockets and database tables similar to files. A short description follows in chapter 3.5.
To minimize the dead time of the HP-RT readout cycle it is possible to write the data into a TCP/IP stream instead of a file. This stream is opened if the VME system has an attached data handler host at run queue start - set by the VME_DHST parameter - and closed at run queue stop. A mechanism - called signal driven IO [05] - allows the receiver host to read data from the stream buffer and to write this data into a new created or already existing output file. Additional the last n data events can be made available for an event display. N can be set with the global parameter VME_ELOG. The interface for state and event logging is explained in chapter 3.6. On VME side the module control and readout functions are defined in a separate library - vme_lib. This library exports the necessary entry point functions and allows the VME module initialization and readout completly independent from the main program. A detailed description is given in the next chapter.

3.4 VME library - vme_lib

To control a wide range of different VME module types only a place holder array - dev_lst - must be delivered to the libraries entry point functions. A linked list of modules with comparable functionality can now attached to the appropriate array field.

field number	0	1	2	3	4	5	6	7
module type	DMC	ADC	TDC	DIS	DPL	GDU	IOR	HVM

DMC:	direct memory access cards
ADC:	analog to digital converters
TDC:	time to digital converters
DIS:	discriminators
DPL:	(dual) programmable logic units
GDU:	gate and delay units
IOR:	input output registers
HVM:	high voltage modules

It is therefore possible to initialize several module types from one or different configurations. Each time a module definition line is parsed the given number of memory blocks is allocated, appended to the appropriate list field and initialized with the following module parameters as shown in the example below:

[TDC][V767]:	1	module definition line
TDC_ADR:	0x050000	TDC address
TDC_ADM:	0x39	TDC address modifier
TDC_IVC:	0x00	TDC interrupt vector
TDC_ICP:	ICP_DIS	TDC interrupt capability
TDC_CHN:	[8FFF8FFF] [0FFF0FFF] [0FFF0FFF] [00000000]	TDC channel mask
TDC_TRG:	0x1F	TDC reference channel
TDC_RMD:	COMMON_STOP	TDC run mode
TDC_RTP:	AQUISITION	TDC run type
TDC_MTP:	RELATIVE	TDC measurment type
TDC_EDG:	RISE	TDC hit edge
TDC_WIN:	40	TDC hit window
TDC_OFW:	-40	TDC hit window offset
TDC_CMD:	[CMD_00]	TDC command marker
[CMD_00]:	0x1600 0	TDC command

If the configuration line parsing is finished a hardware type check at the given VME address follows. If this check is successful, then the VME module can be initialized. Depend on the initialzation return value the necessary module parameters for the handler control and readout are copied into a global memory buffer - provided by the main program. If the module initialization is finished all allocated memory blocks can now released.

Additional the library provides an entry point function to scan the type and the serial number of all or only one module attached to a VME address -> vme_scn_type. Another entry point function is used to read out all modules listed in a handlers run queue (ADCs, TDCs, ...) -> vme_mod_rdo. Already explained in chapter 3.2. the library exports also the necessary functions for the interrupt service initialization and some underlying parts of the IPC system -> devcpy.

3.5 ODB library - odb_lib

This library containes a line parser, a line writer, a number of converter functions, a state and event log interface and an open database connectivity (ODBC) application interface. The ODBC interface and additional the socket access functions **sgets** and **sputs** allow to handle database tables and sockets in the same way as defined for the standard IO file handling. During the program run time the appropriate function set is called. With an additional format parameter in **dopen** it is also possible to create database tables with more then only one column and different column types. A more detailed description is given in (D).

3.6 State and Event Log Interface

Optional event data and/or state information can be written in parallel into a file or a database table. This can be enabled or disabled with the global paramters VME_ELOG for event logs and VME_SLOG for state logs.
If an event display runs at the data receiver host, the IPC system - shown in B.2. - signals each new event. The event display acts here as a server process and is responsible to create, to initialize and finally to release a shared memory device.

3.7 Event Display

The event display runs independent from the readout system and presents in the default online mode each new data event.
Additional it is possible to process raw data files (also the files taken with the LeCroy TDCs) in offline mode (simulation). In both modes the display process can be suspended and later continued.
Two types of event counters are available - a relative counter that can be reset at any time and an absolute counter.
A short program description is given in Appendix C.

Chapter 4

Conclusion and Outlook

With the new drift chamber hardware and software release it is now possible to do the next step and make it available as a lab experiment for students. Before this can be done it is necessary to determine the DRC working point again - the operation conditions at which the chamber provides the highest detection probability (optimal anode high voltage versus optimal ASD-8 thresholds).
Because the TDC type was changed, also a new t_0 calibration is necessary.
The new state and event log interface allows to use a stable and independent working event display, the extended IPC system to integrate the data aquisition control into a more userfriendly environment.
At this place my thank to all who participated in building up and testing the DRC.

Appendix A

A.1 Channel Map and Geometry

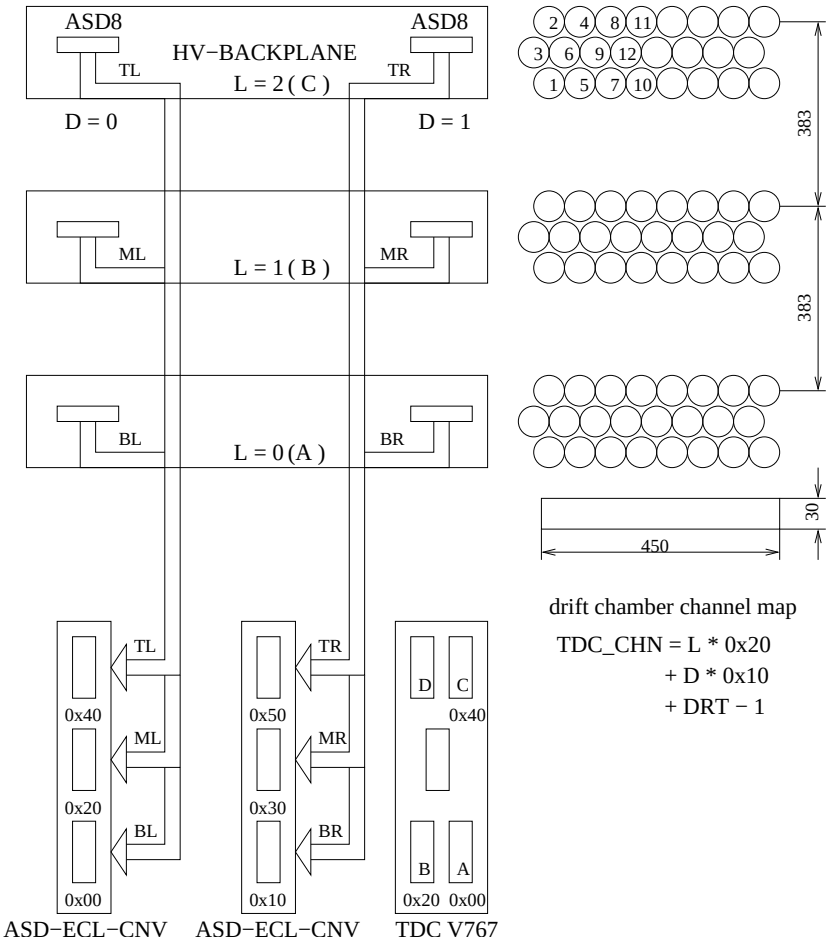
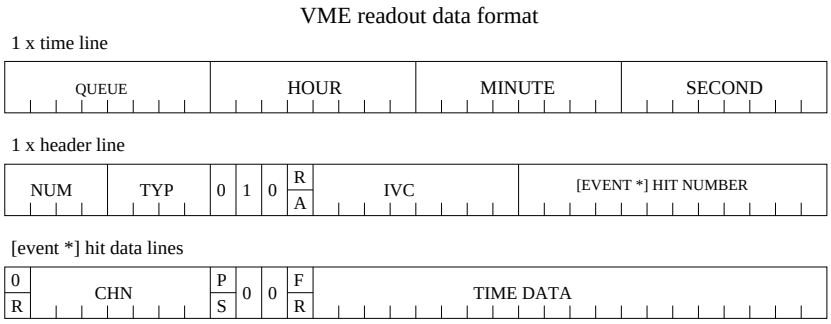


Figure A.1: Channel Map and Geometry

A.2 TDC Readout Data Format



header line
NUM : module number
TYP : type number [1 = TDC_1176, 2 = TDC_V767]
R / A : 0 = relative time measurement, 1 = absolute time measurement [only TDC_V767]
IVC : interrupt vector

data line
0 / R : 0 = data channel, 1 = reference channel [only TDC_V767]
CHN : channel number
P / S : 0 = P [common stop], 1 = S [common start]
F / R : 0 = F [falling edge], 1 = R [rising edge]

Figure A.2: TDC Readout Data Format

A.3 VME Electronics

		Module Typ	Typ	Manufacturer
1	CPU	VME-CPU HP 9000	743	HP
1	HVM	High Voltage Module	VHQ 205L	ISEG
1	DIS	8 Channel Discriminator	V258	CAEN
1	DPL	Dual Programmable Logic Unit	V495	CAEN
1	GDU	8 Channel Gate and Delay Generator	V486	CAEN
1	IOR	16 Channel Programmable I/O Register	V513	CAEN
4	ENC	8 Channel NIM-ECL/ECL-NIM Translator	V538 A	CAEN
1	TDC	128 Channel General Purpose Multihit TDC	V767	CAEN
2	AEC	ASD-8-ECL Translator	AEC	DESY-Z EW
1	COC	Coincidence Unit / PMT Control Supply	COC	HU/DESY-Z EW
1	DIB	ASD-8 Distribution Board	DIB	HU EW

Appendix B

B.1 VME Module Parameters

Each VME module configuration contains common and special parameters. The module addresses and the modules address modifiers are necessary for each module. The interrupt level and vector must be set for interrupt generator modules (IOR). For readout modules (ADCs, TDCs, ...) the interrupt vector value defines the run queue number shared with an interrupt generator module. An interrupt controlled readout is only possible if exactly one interrupt generator module is defined per queue. Optional one or more (maximum 7) readout modules can be attached to a queue. The initialization order is free.
The channel number is only valid for channel providing modules.
A run mode is set module specific and defineded in the structure spc_lst [vme_m.h].

common parameter	description
<MOD>_ADR	VME address
<MOD>_ADM	VME address modifier
<MOD>_INT	VME interrupt level
<MOD>_IVC	VME interrupt vector
<MOD>_CHN	module channel number
<MOD>_RMD	module run mode

All other parameters are module attached special parameters. The special parameter definitions contains the structure mod_lst [vme_m.h] - the special parameter settings the structure spc_lst [vme_m.h].

B.2 IPC Schemes

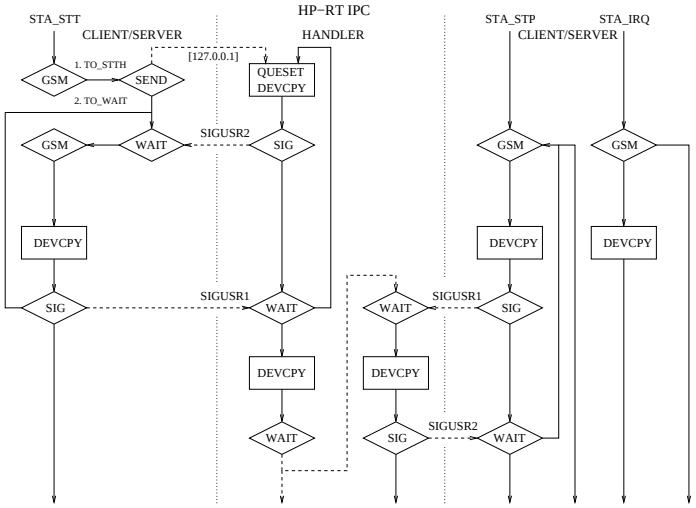


Figure B.1: HP-RT IPC

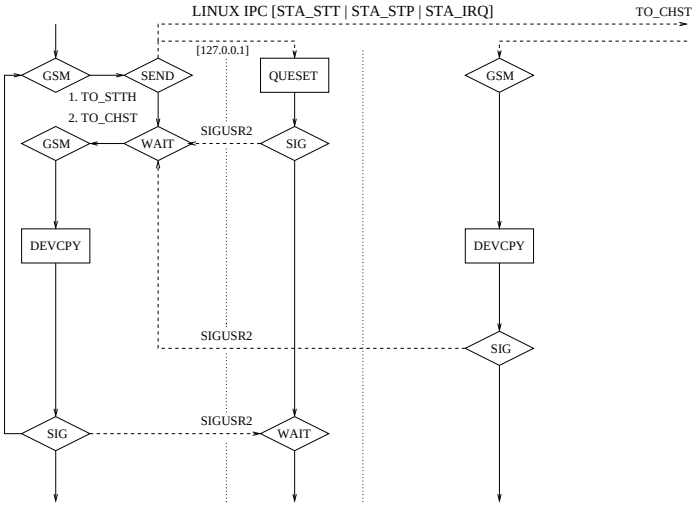


Figure B.2: LINUX IPC

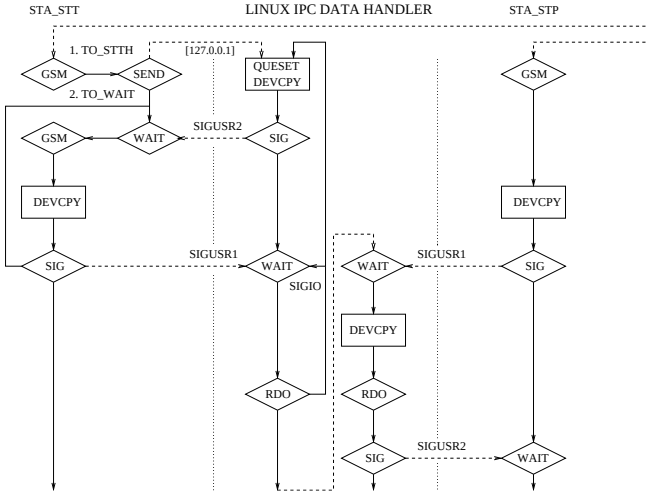


Figure B.3: Data Server IPC

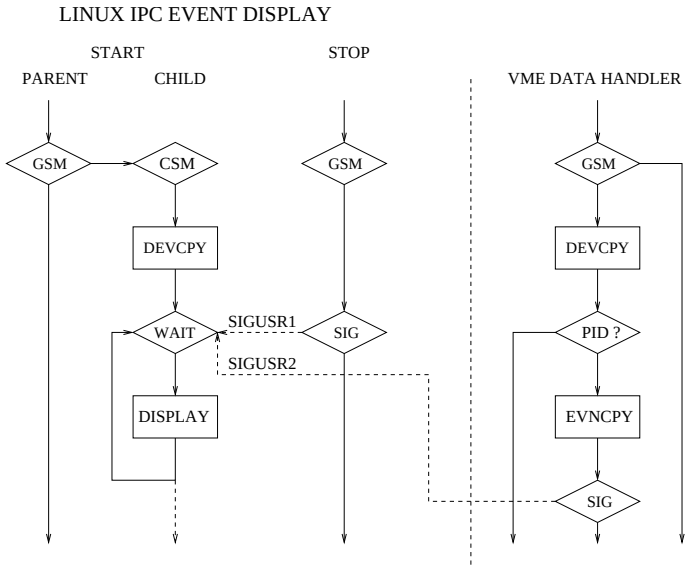


Figure B.4: Event Display IPC

Appendix C

Drift Chamber Users Manual

C.1 Common Remarks

The main program is processed in several run modes - depend on call parameters - on two operation systems - HP-RT and LINUX.
The user interface - the client - is typical a symbolic link to the main program vme_s. It is possible to create links with different names and locations in the file system - for example in $HOME/bin.
The default location of the main program is /usr/local/sbin/vme_s,
the default location of the user interface is /usr/local/bin/vme_c.
On both operation systems the same program parameter set is available - except the database functionality at HP-RT side.

C.2 Global Parameters

Only the parameter set read in by the client - the user interface - is valid and transfered to the clients peer servers.
The default parameter configuration file is /etc/VME/vme.cfg - but also a user defined configuration can be delivered with the program call:

vme_c -g <par_glb.cfg> | <"DB:par_glb"> -s <par_dta.cfg> | <"DB:par_dta">

parameter	description		default value	HP-RT	LINUX
VME_ODBC	ODBC initialization file		/etc/odbc.ini	-	x
VME_ODSN	ODBC data source name		[vme]	-	x
VME_ALLW	hosts allow file		/etc/hosts.allow	x	x
VME_CHST	command execution host		localhost	x	x
VME_WHST	WWW server	**)	localhost	x	x
VME_WPRT	WWW port	**)	80	x	x
VME_DHST	data receiver host		localhost	x	x
VME_PDTA	module parameter data	*)	NULL	x	x
VME_RDTA	raw data output file	*)	vme_dta.raw	x	x

parameter	description		default value	HP-RT	LINUX
VME_ELOG	event log enable/disable		DISABLE	x	x
VME_EDTA	event log file/table		vme_evn	x	x
VME_SLOG	state log enable/disable		DISABLE	x	x
VME_SDTA	state log file/table		vme_sta	x	x
VME_LOGD	log output directory	***)	/etc/VME/log/	x	x
VME_PARD	module parameter directory	**)	/etc/VME/par/	x	x
VME_PRGD	program directory	**)	/etc/VME/prg/	x	x
VME_RAWD	raw data directory	***)	/etc/VME/raw/	x	x

*) can or must be set with command parameters -> program parameters
**) not used in the actual version
***) must be adapted system specific

C.3 Program Parameters

A client call at a terminal console without any additional program parameters lists all available parameters as an usage information.
Only the parameters without a beginning * character are user parameters!

C.3.1 File / Table - Copy

vme_c -a <file> | <"database:table"> -w <file> | <"database:table">

appends the content of a file or database table to an already existing file or table or creates a new file or table and copies the content.
At HP-RT side only the file copy functionality is available.
A call **without** the second parameter -w (or wrt) **deletes** the file or table given by -a (or app).
To select a database and a table the construct <"database:table"> has to be set.
All other necessary values are initalized with settings delivered in VME_ODBC and VME_ODSN.

C.3.2 Raw Data Transfer

vme_c -y <source_file> -b <destination_file>

transfers the content of the source file from host A to host B - given by VME_CHST.

C.3.3 Readout Control

All necessary steps to initialize and start a run queue can be done with only one command call, several module specific calls or module combined calls.
The module initialization order is fixed by an array index defined in vme_lib as listed below.

array index	DMC	ADC	TDC	DIS	GDU	IOR	HVM
	DMC[0]	ADC[0]	TDC[0]	DIS[0]	GDU[0]	IOR[0]	HVM[0]
	DMC[1]	ADC[1]	TDC[1]	DIS[1]	GDU[1]	IOR[1]	HVM[1]
	DMC[7]	ADC[7]	TDC[7]	DIS[7]	GDU[7]	IOR[7]	HVM[7]

The initialization order starts at index 0 (DMC) and ends with 7 (HVM). Only the allocated and initialized memory blocks - beginning from block 0 - are used for the finally module initialization.
If the VME module initialization is finished all allocated blocks are now released.

In the following example a module combined initialization is shown.

Trigger Initialization

To initialize the trigger it is necessary to set the variable delay length DLA and DLB of a gate and delay unit - CAEN V486 - and to program the lookup tables of a logical unit - CAEN V495.
The parameter configuration file or table is delivered with:

vme_c -s vme_c_trg.cfg or vme_c -s "vme:vme_trg"

Each time the VME-CPU has booted or rebooted the trigger initialization is necessary.

Start Queue

A readout queue includes the initialization of the TDC CAEN V767 and the IOR CAEN V513. Optional it is possible to change the raw data output file name.

vme_c -s vme_c_tdc.cfg or vme_c -s "vme:vme_tdc"
vme_c -s vme_c_ior.cfg or vme_c -s "vme:vme_ior" < -b vme_dtb.raw >

As response to a run queue start the state is printed at the clients console:

vme_c: irq[0]: run mod[0]: 1 ivc[0]: 0x00
vme_c: irq[0]: run mod[0]: 2 ivc[0]: 0x00

Stop Queue

A running readout queue can be stopped with

vme_c -x <queue> or vme_c -x all

or suspended with

vme_c -d <msa_adr>

and reinitialized with an appropriate start command.

The <msa_adr> parameter is the most significant address of the TDC or IOR and can be listed with a type query call

vme_c -t all

If one or all queues are stopped the state is printed at the console:

vme_c: irq[0]: stp

State Query

A run state query with

vme_c -q <queue> or vme_c -q all

prints the same state information for one or all queues as shown above.

Set and Reset the High Voltage

The HV module configuration to switch on the anode high voltage can be delivered with following start command calls:

vme_c -s vme_c_hvm.cfg or vme_c -s "vme:vme_hvm" - to set

To shutdown or change the high voltage a different configuration must be used:

vme_c -s vme_0_hvm.cfg or vme_c -s "vme:vme_h0m" - to shutdown

Optimal values for the high voltage are HVM_VLS = 2200 .. 2400.
To switch off the HV this parameter must be 0!

C.4 Event Display

The event display can run different modes - online and offline - and allows simply to switch between both modes. At program start and each time an offline simulation is finished the program returns into the default online mode.
The GTK+-2.0 based user interface is typical a symbolic link to the server program evn_s - so it is possible to create links with different names and locations in the file system. The default location of the user interface is /usr/local/bin/evn_c.

To start the event display simply type in:

evn_c -s

The program can be closed with EXIT or Alt-F4.
Independent from the run mode it is possible to stop and continue the event presentation.
An offline simulation starts if a raw data file is choosen in the file selection window as shown below. During an offline processing the simulation can additional finished with a right mouse button click -> QUIT.
Furthermore two counters are available. Between the default relative and the absolute counter can be toggled with the middle mouse button. The use of the right mouse button resets the relative counter.

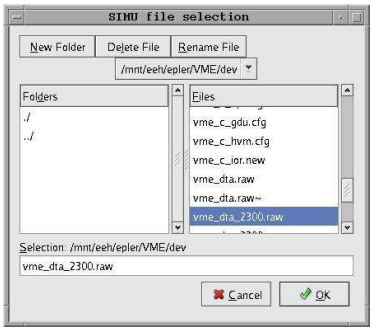


Figure C.1: Offline File Selector

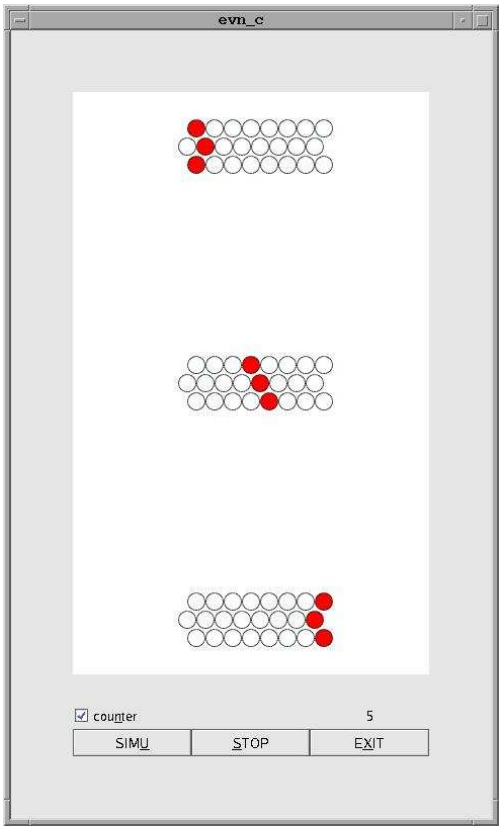


Figure C.2: Event Display

Appendix D

ODBC Application Interface

The ODBC (open database connectivity) API library functions
- dopen, dclose, dgets, dputs, and demove -
are entry point functions to create, to transfer data to and from, and to delete tables in a SQL database system with the same call conventions and return values as defined for the ANSI-C stdio functions
- fopen, fclose, fgets, fputs, and remove.
Additional defined **dopen** C style format parameters can be used and listed at end.

Requirements for installation and database access :

I. (My)ODBC and IODBC libraries and includes must be available
II. all necessary configurations and permissions to connect the database system, to create or delete tables, and to insert data into a table

Entry Point Functions and Format Parameters

FILE * dopen(const char * db_tb_name, const char * format)

db_tb_name:	a string construct "database:table"			
	the double colon is the delimiter to separate database and table			
format:	a format string similar used for the stdio function fopen:			
	"r"	or	"r+"	to open an existing table for read operations
	"w"	or	"w+"	to create a non existing table with one column type varchar
				to open and clear an already existing table for write operatio
	"a"	or	"a+"	to open an already existing table or to create it new for appe

An optional index parameter i is defined to create more then only one column
and to define different column types:
dopen("<database>:<table>", "a+i[int_index %i string_index %20s hex_index %x]");
navigates into the database and creates a table with following index specifiers and column formats:

int_index	int
string_index	varchar(20)
hex_index	int unsigned

If an index specifier is not set or left free an internal format converter creates this as followed:

IDX	for a table with only one column,
IDX_0, IDX_1, ...	for tables with more then only one column

int dclose(FILE * dfp)

dfp: a FILE pointer returned by a dopen call before

The function dclose disconnects from a database system if dfp is a valid pointer - the return value of a successful dopen call - and frees the internal memory allocated by dopen before.
dclose returns always 0.

char * dgets(char * buf, int size, FILE * dfp)

buf: a pointer to a delivered buffer
size: the buffer size
dfp: a FILE pointer returned by a dopen call before

dgets reads one row from a table,
copies the ASCII converted and space separeted values into the buffer and increments an internal read pointer.
If a compare string was set before with
strcpy(ofp->sql_cps, "<index_specifier>=<specifier>");
dgets reads out only the matching parts.
dgets returns the pointer to buf as long as the last row isn't reached or else NULL.

int dputs(char * buf, FILE * dfp)

buf: a pointer to a delivered buffer
dfp: a FILE pointer returned by a dopen call before

dputs inserts the - column type converted - value(s) from the buffer into a table - missing values are written as null equivalents.
dputs returns 0 at success or else EOF.

int demove(const char * db_tb_name)

db_tb_name: a string construct "database:table"

demove connects the database system and navigates into the database,
removes the table - if the table exist and the necessary grants are present - disconnects from the database system, and
returns 0 at success or else EOF.

format parameter	SQL type	
%s	varchar(SQS_MAX)	(SQS_MAX = 254)
%<n>s	varchar(<n>)	(n[1..254])
%c	char	
%i	int	
%x	int unsigned	
%f	double	
%d	date	
%t	time	

Bibliography

[1] Siddeequah Azmi, *Calibration of Cosmic Ray Tracker*, (HU Dep. of Physics June 2005)

[2] The ATLAS Myon Collaboration, *ATLAS muon spectrometer technical design report*, (CERN/LHCC 117-150, May 1997)

[3] HERA-B Outer Tracker Group, *The Front-End Electronics of the HERA-B Outer Tracker Detector*, (DESY April 2002)

[4] H.-U. Kirst, H. Kolanoski, V. Suvorov, F. Tonisch, S. Vassiliev, M. Walter, *Test Results of a Multi-Layer Board Based on the ASD-8B Chip*, (DESY August 1995)

[5] W. Richard Stevens, *UNIX Network Programming* (Prentice Hall PTR, 1998)