EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
ORGANISATION EUROPEENNE POUR LA RECHERCHE NUCLEAIRE

CERN-PS DIVISION

PS/PO/Note 2001-068 (Tech.)

HIGH TENSION CONTROL SYSTEM FOR THE ON-LINE ISOTOPE SEPARATORS - ISOLDE

(with Compact PCI front end controller)

T.Fowler & J.Schipper

Geneva, Switzerland
12 December, 2001

1 Table of contents

2 Introduction

The HT control system for the on-line isotope separator facility ISOLDE is being completely rejuvenated. The former system used a standard "off the shelf" PC running under Windows NT with the interface hardware connected on its local ISA-bus. Severe EMC problems due to sparking of the nearby HT sources, caused a high drop out rate of this system. A further weakness of this system was the interface hardware accessibility and layout.

This system is now replaced by an Industrial CompactPCI system also running under Windows NT, with an open bus hardware architecture. CompactPCI is electrically a superset of desktop PCI (Peripheral Component Interconnect) tailored for use in industrial environments.
The improvements include a rugged Eurocard format, an innovative connector design, greater card capacity, improved immunity to electrical noise, and a more precise, open specification. Together with a growing availability of "off the shelf" interface hardware and a competitive price made us choose the CompactPCI system. First results regarding the above-mentioned EMC problems have shown to be very satisfying.

This note describes the functionality and configuration of the ISOLDE HT control system, in particular the Compact PCI hardware which forms the heart of the control system.

An explanation is given on the installation of the hardware drivers and the DLL's (Dynamic Link Library). These DLL's make it possible to access the Compact PCI hardware via Visual Basic, this is the language in which the equipment control, a Microsoft ActiveX component, is written.

A detailed description of the hardware control functions is given in the Appendix together with a list of the properties and their initialization order and values.

The ISOLDE control model is briefly discussed, more on this can be found in reference 1, however the hardware control part of the ISOLDE model, the equipment control, is discussed in greater detail.

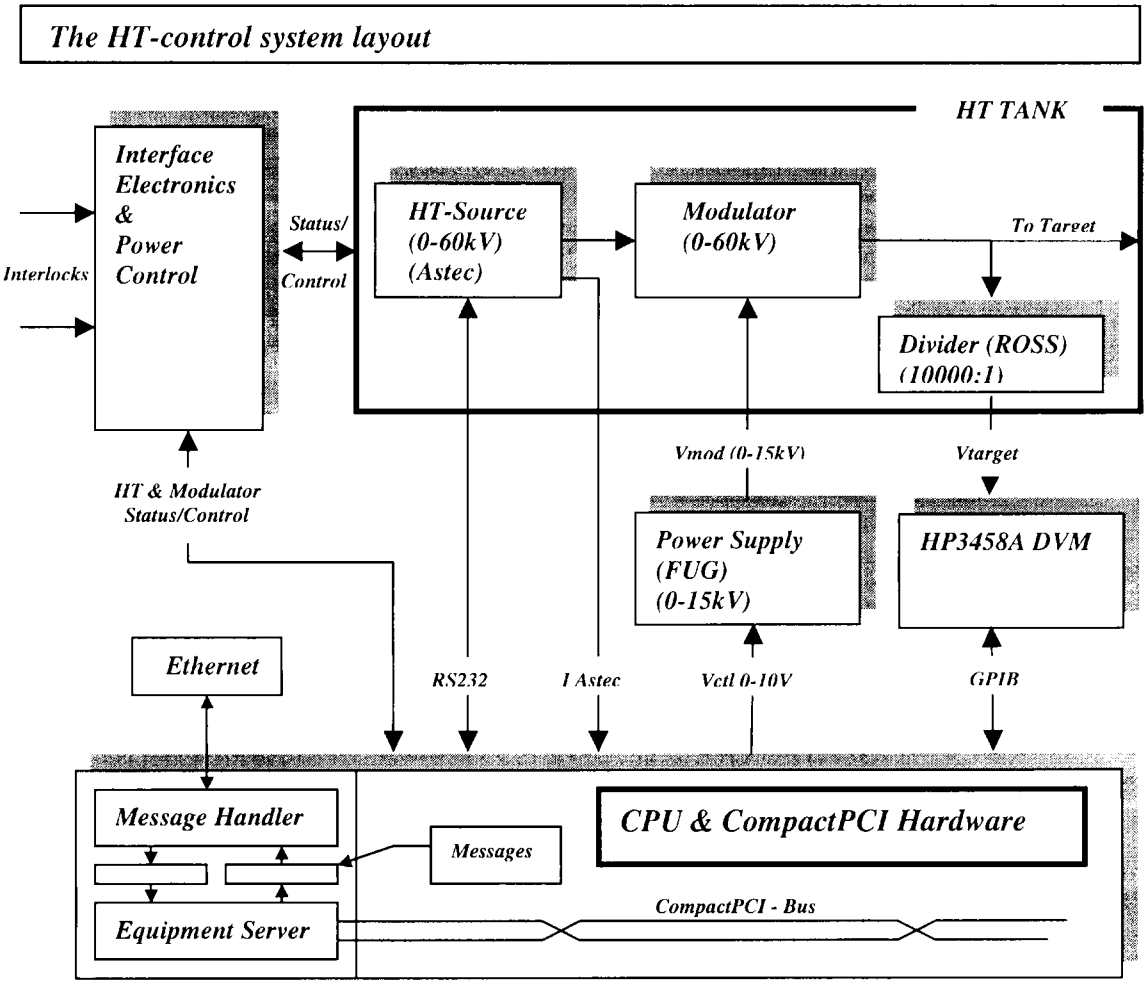
An user interface, the HT-control application program, is available for the HT-control system simplifying direct access to the essential properties.

Furthermore a brief overview of the HT-source and the modulator is given, in order to obtain a global picture of the overall system.

(In this note reference is made to directories and servers containing source code and application programs. Although care is taken to maintain their location, these are always liable to change.)

3 The HT- control system principle

The principal task of the HT-control system is to supply the target potential for ion acceleration and control a modulator for this target voltage. This modulator has the task to fully discharge the target prior to beam impact and to then restore the voltage close to its nominal value.



The HT-source (Astec Very High Power - reference 5), a fast acting high stability hard tube power supply, applied to the target is modulated (if requested) by a resonant circuit whose peak voltage amplitude is determined by an externally controllable power supply.
This power supply is a commercially available (FUG Elektronik - reference 6) 0-15kV, 400 J/s capacitor recharging power supply which operates at approximately 11.6 kV to resonate the target voltage through 60kV. The desired output voltage is obtained by applying an external control voltage. (0 - 10Vin gives 0-15kVout.)
The output of the modulator connected to the target is fed back via a 10000:1 (ROSS Engineering Corporation - reference 7) high voltage divider to a high precision Voltmeter (HP3458A Hewlett Packard .)
(See reference 2 for more details on the modulator principle.)

The ISOLDE ion separator facility consists out of two separators, the General Purpose Separator (GPS) and the High Resolution Separator (HRS).
Both separators have their own HT-source and associated hardware, however all control functions are situated in the same control crate. This crate, the CompactPCI system, forms the heart of the control setup together with the Interface Electronics.
The tasks it has to perform are:

Control tasks:

- HT- source control.
- Modulator control.

Acquisition and Status tasks:

- Read HT-source voltage.
- Read HT-target voltage.
- Read HT-source current.
- Read status HT-source.
- Read status modulator.

3.1 HT-source control

The HT-source is controlled via an opto-isolated RS232 link the control commands are:

	ACTION	**COMMAND**	
-	Set HT- source voltage (0-60kV)	"HV=(value)"	
-	Set HT on	"HV= ON"	
-	Set HT off	"HV =OFF"	
-	Set slope up speed (kV/sec)	"HVSI = (value)"	
-	Set slope down speed (kV/sec)	"HVSO = (value)"	
-	Set Protect mode	"*x1<CR><LF>"	"*x3<CR><LF>"
-	Set NO protect mode	"*x0<CR><LF>"	"*x2<CR><LF>"

('value' allows two decimal places and is in kV.)

HVSI & HVSO are the speed at which the HT can change 'up' or 'down'.
NOPROTECT mode makes the HT-source less sensitive to high voltage sparking, but care has to be taken when used since it increases the possibility of damaging the HT-source.

The commands "HVSI" and "HVSO" return the set slope speeds.
"HV" returns the HT level in kV, with a maximum resolution of two decimal places.

3.2 Modulator control

The Modulator has four basic control commands:
- ON, OFF, STANDBY and RESET.

These commands are sent to the modulator via the Interface Electronics and are negative TTL logic levels.
There is another signal called STROBE, which is used to latch the commands into the Interface Electronics.

The protocol is such that when a command is activated, the STROBE should be activated at the same time and the command should remain active for a minimum of 1msec after the strobe is negated.
The STROBE length should be greater than 5msec.

This way of command latching prevents spurious signals at the command inputs to activate undesired control over the modulator.

- Modulator control voltage for the FUG power supply 0-10V.

This control voltage sets the desired modulator voltage and is a function of the HT-source voltage.

3.3 Acquisition and status tasks

List of acquisition and status tasks:

- The HT-source voltage is read via the RS232 link as mentioned previously.

- The HP3458A DVM reads the HT target voltage, which is first divide by a high voltage divider (10000:1).

This is a high precision measurement with a resolution of up to 6 1/2 digits, depending on the settings of the HP DVM device.
The DVM is completely software controllable via the GPIB port.
The desired configuration can be set via the equipment module.
Furthermore this device can be used for graphing purposes i.e. one can visualize the performance of the HT-source.

- The HT-source current is measured over a 1k ohm resistor in the current monitoring feedback loop of the HT-source. This gives a straight readout conversion value in mA.

- The Status of the modulator and HV-source is returned by the Interface Electronics and contains the following data:

    - OFF
    - ON
    - STANDBY
    - TRANSITION (warming up)
    - REMOTE/LOCAL (control of the modulator)
    - OK (no interlock)
    - MAINS ON (220V present on Interface Electronics)
    - HV PULSING (modulator ON & HV ON)
    - HV STATIC (modulator STANDBY & HV ON)
    - HV ON
    - HV OFF

All these signal are TTL level signals.
The first five status bits refer to the modulator.

4 CompactPCI (CPCI) Hardware

The previously mentioned control & acquisition tasks, results in the following hardware requirements for one separator:

- 1 - RS232 control port for HT-source control.
- 1 - GPIB control port for HP-DVM control.
- 1 - DAC (analog out) port for modulator control voltage.
- 1 - ADC (analog in) port for HT-source current measurement.
- 5 channel output register for modulator control.
- 11 channel input register for status.

4.1 CPCI Hardware selection & configuration

The CPCI rack enclosure selected is a GESPAC CPCI-784SP-8E/U.

CPU (PCISYS-56AE also from GESPAC) + peripheral information:
Pentium MMX 133Mhz
RAM 64Mbyte
Hard disk 3.6Gbyte
Ethernet connections are possible to either structured or non-structured networks.
The open bus architecture allows a connection for 8 CompactPCI 6U Eurocards.

The software platform used is WinNT therefore compatible drivers have to be delivered with the hardware.
For future updates modularity would be recommended, i.e. dedicated cards for the different functions.
This approach is already widely used in the Kicker control systems, with the VMOD-IO MODULbus VMEcarrier boards from JANZ Computer AG .
These carrier boards can hold 4 plug in modules of the MODULbus series.
A similar carrier board exists for the CPCI system, the CMOD-IO MODULbus CPCI carrier board, therefore the MODULbus plug in cards can be exchanged between the two systems.
Since the PS controls group uses the MODULbus plug in cards in a wide variety of applications, this seems to be the appropriate chose.

The selected CPCI MODULbus plug in cards for the complete HT-control system:

- 2 x VMOD-SIO2 (2 independent opto isolated RS232 channels)

Channel	**Function**
0	acquisition of the HT-source
1	Voltage monitoring of the HT-source (future development)

- 2 x VMOD-GPIB (1 channel General Purpose Interface Bus module - 1GPIB/separator)

- 2 x VMOD-8E8 (8 differential multiplexed analog inputs, 8bit-ADC)

Channel	**Function**
0	HT-source current measurement (scale 0-5V, resolution 20nA/bit)

- 2 x VMOD-8A4 (4 unipolair multiplexed analog outputs, 8bit-DAC)

Channel	**Function**
0	Modulator control voltage (scale 0-10V, resolution 40mV/bit)

There is an output relay for disconnecting the analog output from the connectors.

- 2 x VMOD-BE20 (20 channel opto isolated input register)

Channel	**Function**
0	OFF
1	ON
2	STANDBY
3	OK
4	REMOTE
5	TRANSITION
6	MAINS ON
7	HV PULSING
8	HV STATIC
9	HV ON
10	HV OFF

The maximum input voltage of the VMOD-BE20 can be 80V.
The minimum input voltage for high level = 3.2V below this voltage a low level will be read.
(Care has to be taken when TTL is used since TTL has a lower high level threshold)

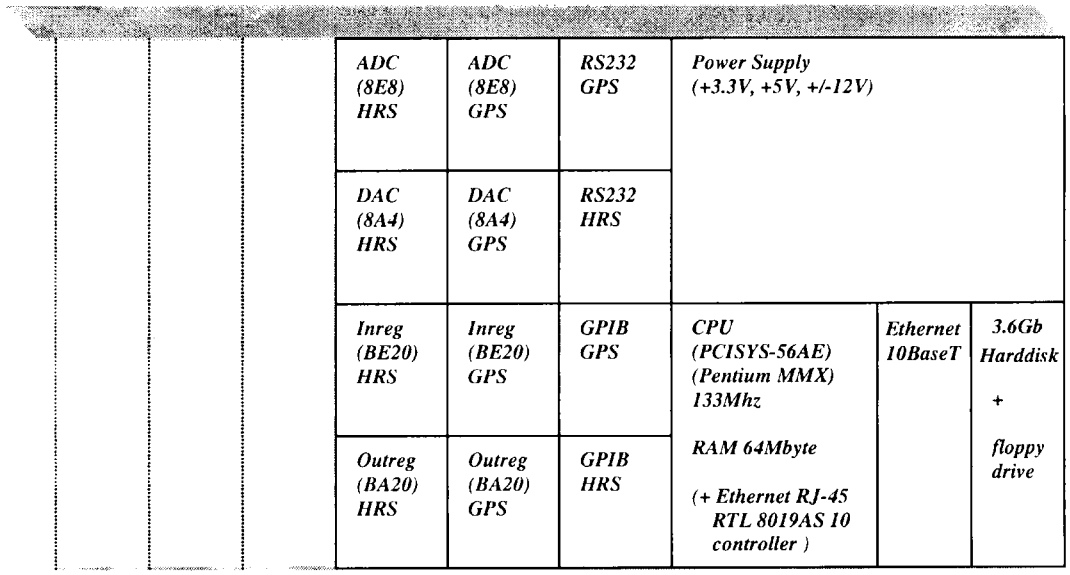
- 2 x VMOD-BA20 (20 channel opto isolated output register)

Channel	Function
0	OFF
1	ON
2	STANDBY
3	RESET
4	STROBE

The maximum voltage that can be supplied to the VMOD-BA20 is 80V with maximum current of 500mA.
The load can be connected to the 'high' or 'low' side of the contacts.
In this case the load is connected to the 'high' side, which causes an inversion of the logic.

These plug in cards require 3 CMOD-IO carrier board to host them.
To keep a separation between the GPS and the HRS system, every system has its own plug-in card.

***CompactPCI control crate layout** (GESPAC CPCI-784SP-8E/U.)*



The VMOD plug in card position should correspond with the software settings in the registry.

5 The ISOLDE control system model

The ISOLDE control system model is very similar to the PS model, which is based on equipment names.
Every equipment is also known as a physical device (e.g. HT power supply), has a unique ASCII name and is registered in the control system database.
Each equipment is member of an equipment type family, called class.
A class describes all parameters that can be controlled for this type.
Every parameter is identified by an ASCII name, called property. The implementation of a class and its properties is carried out in an equipment module, residing in the Front End Computer. (FEC)
These FEC's form the link between the application programs running on the consoles and the equipment.

The principle of these FEC models are described in great detail in reference 1 in this note we will concentrate on the equipment module or better the equipment control where the equipment specific properties are handled.
These equipment controls are ActiveX Controls and are written in Visual Basic.

The class name for the HT control is HTCONTROL and the name of the FEC is HTFEC.
The HTFEC contains the control and acquisition of two equipments :

- GPS.HTCTL
- HRS.HTCTL

These two equipments one for the General Purpose Separator (GPS) and one for the High Resolution Separator (HRS), are a subset of the class HTCONTROL.

The properties of the class HTCONTROL can be found in Appendix B. This is basically a copy of the ISOLDE database (\\SRV5_NICE\PGM\P32\CONTROL\ControlDB\Pgm\ControlDB.exe).

5.1 Hardware control from equipment module

The equipment control written in Visual Basic communicates with the hardware via their dedicated drivers. This can not be done directly from Visual Basic but is done via the so-called DLL's.
A DLL, Dynamic Link Library, is a library of routines that are callable at runtime from any application that conforms to the Windows API. The DLL's for the CPCI hardware are written and compiled in MS Visual C++ 5.0.

Every hardware module has a DLL with a set of functions adapted to its capabilities. Every function that is used from the DLL has to be declared in a Visual Basic module. These functions are addressed by their so-called decorated names this is the name of the function after compilation. The compiler prefixes the function name with an underscore (_) and appends the function name with an at sign (@) character followed by the number of bytes in the argument list (the required stack space).
For the DLL's to be called from the Visual Basic program an 'alias' clause should be added to the **declare** statement. An example of this can be found in the source code of the HTCONTROL equipment (see below).

In Appendix D an overview is given of the used DLL's with an explanation of the functions and there arguments.

5.2 Description of the equipment control.

(To have a better understanding of this program it is advised to read reference 1.)

The source code for the HTCONTROL equipment control can be found on the ISOLDE server \\SRV1_ISOLDE\CTL\FecNT\EQPOBJ\HTCONTROL .

The main program for all FEC's is the FECMAIN program, it is a generic program and encapsulates the HTCONTROL.ocx equipment control component.
FECMAIN also loads the EQPINIT.ocx and RPCSRV.ocx generic components, the first one is responsible for the initialization sequence the latter serves the network requests.

Sequence of actions after start of FECMAIN

- Initialization phase
  The properties are initialized in the order and with the values given by the database (Appendix C.)
  - Set the RS232 com port
  - Configure the RS232

- Set the hardware addresses
- Configure the HP-DVM
- Set data logging parameters
- Set scaling factors for current measurement, HT target voltage measurement and modulator control voltage.
- Set limits HT-source
- Initialize the CPCI hardware
- Initialize the HP-DVM
- Send slope settings (HVSI & HVSO) to HT-source
- Send reset to control interface (reset interlocks)
- Restore last values for properties CCV(demand voltage) and TRIG (trigger setting for HP-DVM)

The properties are always defined by two methods, one to read (get) and one to write (let). Writing and reading the properties, handled by the RPCSRV.ocx component is relatively simple, as far as the equipment control is concerned. The requested part of the code is executed and data returned or set.

The acquisition and status data are refreshed every 5 seconds. This is done with the EqpMain_Callback subroutine.
The call back time is set in USERControl_Initialize and could be changed to obtain a higher refresh rate.

- Callback phase activities

  - Store errors from last call back for debugging purposes
  - Read HT-source voltage via RS232
  - Read HT-source current
  - Read status modulator
  - Read status HT-source
  - Configuration and read-out of the HP-DVM via GPIB for the target voltage (more details later)
  - Data logging

The reading of the status and HT-source voltage and current are relatively straight forward, however the high precision reading/measurement of the target voltage requires further explanation.

5.3 The HP-DVM control routine

The HP-DVM is controlled via the GPIB protocol and measures the HT-target voltage, it serves two tasks :

1. Standard set-up
   Makes an average of a series of high precision measurements, the configuration is defined in the equipment module. (properties APER1, NRDGS1, DELAY1, TIME1)
2. Graphing set-up . (properties APER2, NRDGS2, DELAY2, TIME2)
   Allows visualization of the HT-target voltage (resonance, recovery and ripple)

Care has to be taken when controlling the HP-DVM for several reasons (the main ones being):
- When HP-DVM is busy (waiting for a trigger) it can not be interrupted i.e. if it doesn't receive a trigger, it remains in the busy status.
- GPIB controller has to be able to handle measurement speed.
- Configuration of the properties APER, NRDGS, DELAY, TIME have to be carefully checked to prevent the HP-DVM from blocking.

The HP-DVM must be prevented from blocking at any time, this is done by 'serial polling' the device in every callback, if after a set number of callbacks the HP-DVM is not replying the device will be reset.
Serial Polling - The process of polling and reading the status of one specified device on the bus.

In Appendix E a flowchart for the HP-DVM control routine is shown.

5.4 Data logging

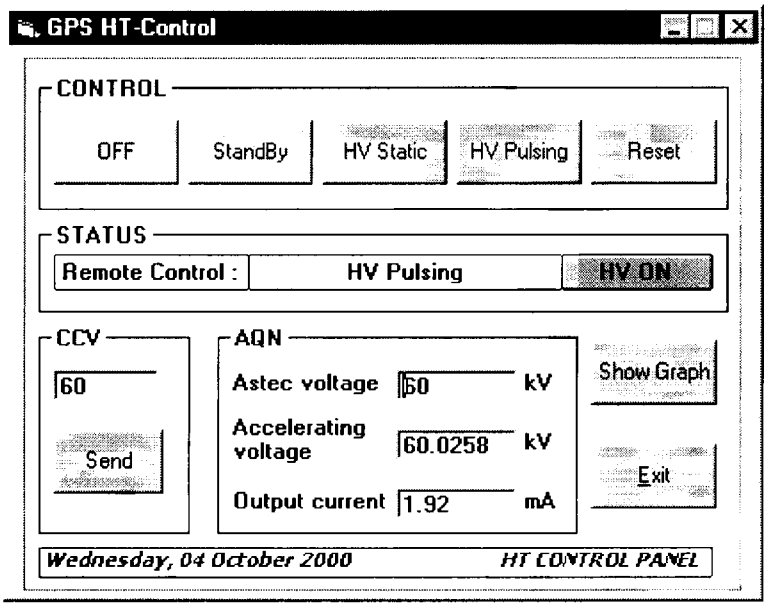
Every predefined time, property LOGTIME (in minutes), a sample is made of some of the properties and stored in a log file. The log file path is defined by the property LOGFILE.
Every month a new file is opened containing the following data :

- Log time
- Requested HT-source voltage - CCV
- HT-source voltage - AQN
- HT-target voltage - AQN1
- HT-source current - AQN2
- Status modulator - STAQ
- Status HT-source - STAQ1

6 Application programs

6.1 The HT control application program

To allow simple access to the frequently used properties of the HT-source & modulator, an application program, written in Visual Basic, is available for the users.
\\SRV1_ISOLDE\CTL\CONTRL32\HT (shortcut GPS.HT & HRS.HT)



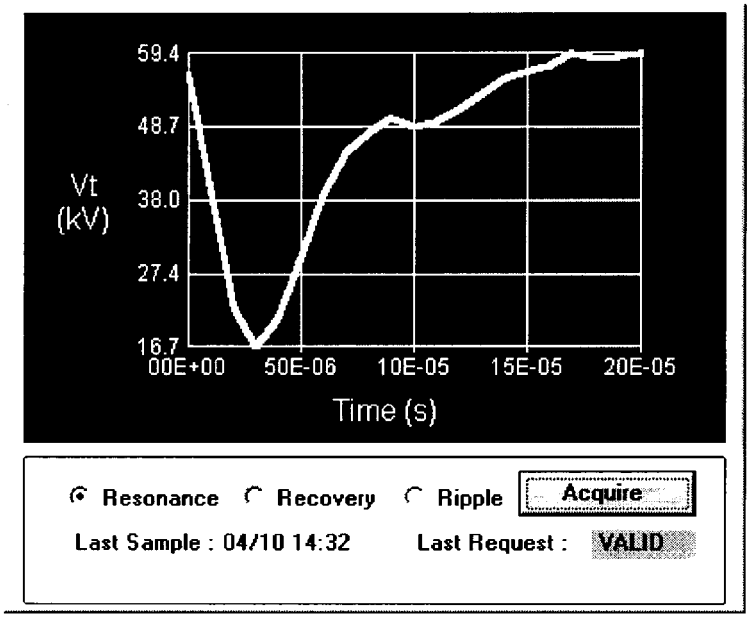
Panel Layout of HT-control application program.

The five control buttons have the following functions:

OFF	- Modulator OFF, Astec OFF
Standby	- Modulator in standby (timing blocked), Astec OFF
HV Static	- Modulator in standby (timing blocked), Astec ON
HV Pulsing	- Modulator ON, Astec ON
Reset	- Reset Interlocks

When NO PROTECT mode is active the HV ON label changes to HV ON [NP].

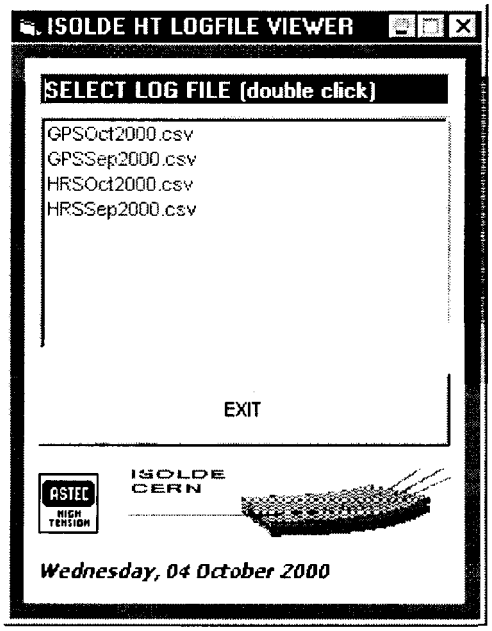
The **Show Graph** control button makes it possible to visualize the HT behavior, below one can see an example if the option resonance was selected.



The Show Graph Option

6.2 Read Log file application program

A Visual Basic program is available to read the log files, the program opens the files in the EXCEL environment and decodes the status words.
To use this routine, simply copy the shortcut **IsoldeLog** from:
\\SRV1_ISOLDE\CTL\HOME\FecHT\LOGHT\DataLogISO\PGM\ to your desktop.



Panel Layout of Logfile viewer.

7 References

1. Windows NT as Device Server for the ISOLDE-REX Project - I.Deloose
   PS/CO Note 97-27.

2. A 60 kV Modulator for the target voltage of an on-line isotope separator
   D.C. Fiander, A.Fowler
   CERN-PS.

3. HP 3458A Multimeter Operating, Programming and Configuration Manual.
   Hewlett Packard Company.

4. Hardware manual MODULbus-series by JANZ COMPUTER AG
   - CMOD-IO Compact PCI MODULbus carrier (Version1.0)
   - VMOD-SIO2 MODULbus serial I/O (Version1.3)
   - VMOD-GPIB IEEE-488 Interface module (Version1.0)
   - VMOD-8E8 MODULbus analog input (Version2.1)
   - VMOD-8A4 MODULbus analog output (Version1.1)
   - VMOD-BE20 MODULbus digital input (Version1.0)
   - VMOD-BA20 MODULbus digital output (Version1.0)

5. Astec 60kV Precision Power Supply
   Circuit diagrams & Operator's manual
   ASTEC VERY HIGH POWER England

6. Capacitor Charging Power Supplies HCK-Series
   Operating Instructions and Description
   F.u.G. Elektronik GmbH.

7. VD120-12.5-BD-K-B High Voltage Divider Calibration Chart
   ROSS Engineering Corporation.

A Appendix A.

A.1 Software installation procedure for the CMODIO hardware.

(A copy of all files mentioned can be found in:
\\SRV1_ISOLDE\CTL\HOME\FecHT\InstallPCImod
(Copy this directory to the C:\ drive of the CompactPCI system)

1. Install Carrier CMODIO
   Run cmodio_b.exe in c:\InstallPCImod\carrier
   (next - next - select driver only - next - next - finish)
   don't reboot system yet -> Cancel

2. Install plug-ins c:\InstallPCImod\Plug-ins

   a. Install DAC VMOD-8A4 run: m8a4_1_0b.exe
   (next - select driver only - next - next - next)
   ignore error message if present, this is for the registry of the modules and will be done at the end of the installation.

   b. Install ADC VMOD-8E8 run: m8e8_1_0b.exe
   (next - select driver only - next - next - next)
   ignore error message if present, this is for the registry of the modules and will be done at the end of the installation.

   c. Install VMOD-BA20 output register run: mba20_b.exe
   (next - next - select driver only - next - next - finish)
   don't reboot system yet -> Cancel

   d. Install VMOD-BE20 input register run: mbe20_b.exe
   (next - next - select driver only - next - next - finish)
   don't reboot system yet -> Cancel

   e. Install VMOD-GPIB card run: mgpib_1_0b.exe
   (next - select driver only - next - next - next)
   ignore error message if present, this is for the registry of the modules and will be done at the end of the installation.

   f. Install RS232 card VMOD-SIO2
   double click rs232.reg in dir c:\InstallPCImod\Sio2-Rs232
   This enters all the required settings in the registry, and is all that has to be done for the RS232 plug-ins.

3. Install DLL's.
   The required DLL's can be found in dir C:\InstallPCImod\Dll all these DLL's have to be copied to C:\Wnt\system32\

4. Reboot the PC now.

5. Now the plug-in cards have to be connected software wise to the carrier boards, this of course in correspondence with the hardware set-up.

These software settings have to be entered in the registry.
This is done with the help of the ModulBusNTconf.exe program, which can be found in C:\InstallPCImod\RegHardware.

Run this program and add the desired plug-in module.

Example for VMOD-GPIB plug-in:

[Select] ADD module

Type	[Select GPIB]	
Number	0	for the first GPIB module the next one would be 1
Carrier Name	CMODIO	No 0 (if module is plugged in on the first carrier board)
Slot	2	(position on carrier board - top slot is 0)

Repeat this operation for all modules that are used in the set-up.
Note that numbering always starts at 0.
When finished save the changes in the registry.
(For the RS232 module this is already done by double clicking rs232.reg in directory C:\InstallPCImod\Sio2-Rs232 see 2f)

6. Reboot once again, after reboot the modules are ready to be used.

B Appendix B

B.1 Property list

Property Name	Access	Size	Description
APER1	R/W	DOUBLE	Measurement aperture
APER2	R/W	DOUBLE	Measurement aperture - graphing module data
AQN	R	DOUBLE	Read physical Astec voltage
AQN1	R	DOUBLE	Read physical target voltage - averaged value
AQN1TIME	R	TEXT	Read AQN1 data acquisition-time stamp
AQN2	R	DOUBLE	Read physical current
AQN3	R	DOUBLE	Voltage acquisition of samples array for graphing
BAUD	R/W	LONG	Baudrate
BIT	R/W	BYTE	Number of data bits
CCV	R/W	DOUBLE	Set physical demand voltage
COMPORT	R	BYTE	Com port number
CYCLETIME	R/W	LONG	Specifies the maximum possible interval between regular timing pulses (assuming one ISOLDE user programmed per PSB supercycle not used at present)
DELAY1	R/W	DOUBLE	Post-trigger measurement delay
DELAY2	R/W	DOUBLE	Post-trigger measurement delay - graphing module
ERRORTEXT	R	TEXT	General error description
ERRORTEXT1	R	TEXT	General error description
EVEN	R/W	BYTE	Parity even=TRUE, Odd=False
GPIBC	R/W	SHORT	GPIB Controller card address
GPIBI	R/W	SHORT	HP3458A GPIB card address
GRAPHREQ	R/W	BYTE	Request for graphing
GRAPHERROR	R	BYTE	Error in graphing request
HVSI	R/W	DOUBLE	Voltage slope in
HVSO	R/W	DOUBLE	Voltage slope off
INIT	R	BYTE	Initialize the hardware
INIT1	R	DOUBLE	Initialization.
IOCARDOFFSET	R	SHORT	I/O card offset address
LOGFILE	R/W	TEXT	Data logging file path
LOGTIME	R/W	LONG	Sets the voltage data logging frequency in minutes
MAXS	R/W	DOUBLE	Maximum voltage slope for Astec HVSI and HVSO
MAXV	R/W	DOUBLE	Maximum demand voltage
MINV	R/W	DOUBLE	Minimum demand voltage
NRDGS1	R/W	SHORT	Number of readings
NRDGS2	R/W	SHORT	Number of readings - graphing module
PARITY	R/W	BYTE	Parity on=TRUE else False
SCLRI	R/W	DOUBLE	Scaling factor for current read
SCLRV	R/W	DOUBLE	Scaling factor for voltage read
SCLW	R/W	DOUBLE	Scaling factor for FuG p.s.u. voltage write
STAQ	R	DOUBLE	Status Acquisition for Resonator (coded, see below)
STAQ1	R	BYTE	Status Acquisition for Astec (coded, see below)
STCC	R/W	DOUBLE	Status Control for resonator (coded, see below)
STCC1	R/W	DOUBLE	Status Control for Astec (coded, see below)
STDERROR	R	BYTE	No trigger detection

STOPBIT	R/W	BYTE	Number of stop bits
TARM	R/W	BYTE	Trigger arming
TEST1	R/W	LONG	Test property
TEST2	R/W	LONG	Test property
TIME1	R/W	DOUBLE	Sampling time interval
TIME2	R/W	DOUBLE	Sampling time interval - graphing module
TRIG	R/W	BYTE	Trigger configuration

IOCARDOFFSET is a reference for the address of the CMODIO plug-in cards.
Every plug-in installed has a number (starting at 0), every plug-in of the same type has an incremented number, since per HT system there is only one plug-in of its kind installed, we can address the hardware directly with this property. In this case 0=GPS and 1=HRS.
This way of address decoding minimizes the number of properties and is possible since all information concerning the hardware configuration is present in the registry.

(Care has to be taken with the I/O modules VMOD-BA20 and VMOD-BE20 since these modules return file descriptors when opened, which then have to be used for further handling).

B.2 Property bit significance

STAQ (Status & Acquisition Modulator)

1	-	OFF
2	-	ON
4	-	STANDBY
8	-	INTERLOCK
16	-	LOCAL
32	-	WARMING UP
64	-	NO 220V
128	-	HV STATIC
256	-	HV PULSING

STAQ1 (Status & Acquisition Astec)

1	-	ASTEC OFF
2	-	ASTEC ON
4	-	ASTEC FAULT
8	-	PROTECT
16	-	NO PROTECT

STCC (Control Resonator)

1	-	OFF
2	-	ON
4	-	STANDBY
8	-	RESET

STCC1 (Control Astec)

1	-	ASTEC OFF
2	-	ASTEC ON
4	-	PROTECT
8	-	NO PROTECT

TARM (Trigger Arming HP)

1	-	AUTO
2	-	EXT

TRIG (Trigger Configuration HP)

1	-	SGL
2	-	EXT

C Appendix C

C.1 Property initialization order

Initialization sequence	GPS.HTCTL	HRS.HTCTL
COMPORT	10	12
IOCARDOFFSET	0	1
BAUD	9600	9600
BIT	8	8
EVEN	0	0
PARITY	0	0
STOPBIT	1	1
GPIBI	22	22
GPIBC	0	0
APER1	0.001	0.001
APER2	0.000002	0.000002
DELAY1	0.05	0.05
DELAY2	0	0
NRDGS1	100	100
NRDGS2	200	200
TIME1	0.005	0.005
TIME2	0.00001	0.00001
LOGFILE	O:\HOME\FECHT\LOGHT\GPS	O:\HOME\FECHT\LOGHT\HRS
LOGTIME	10	10
SCLRI	1	1
SCLRV	10	10
SCLW	75	75
MAXS	10	10
MINV	0	0
MAXV	62	62
INIT	0	0
INIT1	0	0
HVSI	10	10
HVSO	2	2
STCC	8	8

CCV, TARM & TRIG are set to restore last value.

D Appendix D

D.1 The hardware control functions (DLL's)

A list of all functions (+ decorated name) accessible from the DLL's.

D.2 mgpib.dll

- **MGPIBOpen** - The function returns 0 if the operation was successful, otherwise -1 is returned.

```
_MGPIBOpen@4
long MGPIBOpen(long modno)
long modno,        // module number
```

- **MGPIBClose** - The function returns 0 if the operation was successful, otherwise -1 is returned.

```
_MGPIBClose@4
long MGPIBClose(long modno)
long modno,        // module number
```

- **MGPIBReadRegister** - read a GPIB controller register

```
_MGPIBReadRegister@12
long MGPIBReadRegister(long modno, long regNo, long *result)
long modno,        // module number
long regNo,        // controller register number
long *result       // read result value
```

- **MGPIBWriteRegister** - write a GPIB controller register

```
_MGPIBWriteRegister@12
long MGPIBWriteRegister(long modno, long regNo, long sValue)
long modno,        // module number
long regNo,        // controller register number
long sValue        // value to write
```

- **MGPIBSetLocalAddress** - set local GPIB address

```
_MGPIBSetLocalAddress@12
long MGPIBSetLocalAddress(long modno, long primAddr, long secAddr)
long modno,        // module number
long primAddr,     // primary module address
long secAddr       // secondary module address
```

- **MGPIBDefineEIO** - define a new EOI character

```
_MGPIBDefineEOI@8
long MGPIBDefineEOI (long modno, long c)
long modno,        // module number
long c             // EOI character
```

- **MGPIBSetTimeout** - set a value for GPIB timeout

```
_MGPIBSetTimeout@8
long MGPIBSetTimeout(long modno, long timeout)
long modno,             // module number
long timeout            // value for timeout
```

- **MGPIBSendCommand** - send a command

```
_MGPIBSendCommand@8
long MGPIBSendCommand(long modno, long cmd)
long modno,     // module number
long cmd        // command code
```

- **MGPIBSendString** - send a string to a connected device

```
_MGPIBSendString@16
long MGPIBSendString(long modno, long primAddr, long secAddr, long *buffer)
long modno,     // module number
long primAddr,  // primary address of device
long secAddr,   // primary address of device
long *buffer    // pointer to buffer with data to send
```

- **MGPIBRecvBuffe**r - receive data from a connected device into a buffer returns number of bytes send, if not successful -1

```
_MGPIBRecvBuffer@20
long MGPIBRecvBuffer(long modno, long primAddr, long secAddr, long *buffer, long count)
long modno,     // module number
long primAddr,  // primary address of device
long secAddr,   // primary address of device
long *buffer,   // pointer to buffer with data received
long count      // maximum number of bytes, which can be received
```

- **MGPIBWriteBuffer** - write bytes from a buffer to data out register

```
_MGPIBWriteBuffer@12
long MGPIBWriteBuffer(long modno, long *buffer, long count)
long modno,     // module number
long *buffer,   // pointer to buffer
long count      // number of bytes to write
```

- **MGPIBSendBuffer** - send data from a buffer to a connected device

```
_MGPIBSendBuffer@20
long MGPIBSendBuffer(long modno, long primAddr, long secAddr, long *buffer, long count)
long modno,     // module number
long primAddr,  // primary address of device
long secAddr,   // primary address of device
long *buffer,   // pointer to buffer with data to send
long count      // number of bytes to send
```

- **MGPIBLastError** - this function returns the last error code

```
_MGPIBLastError@4
long MGPIBLastError(long modno)
long modno,     // module number
```

D.3 m8e8.dll

- **M8E8Open** - The function returns 0 if the operation was successful, otherwise -1 is returned.

```
_M8E8Open@4
long M8E8Open(long modno)
long modno,        // module number
```

- **M8E8Close** - The function returns 0 if the operation was successful, otherwise -1 is returned.

```
_M8E8Close@4
long M8E8Close(long modno)
long modno,        // module number
```

- **M8E8ReadChannel** - reads a value from a analog input channel of a VMOD-8E8 module

```
_M8E8ReadChannel@12
long M8E8ReadChannel(long modno, long channel_no, long* result)
long modno,              // module number
long channel_no,         // channel number
long* result             //pointer to result
```

- **M8E8LastError** - List of error codes in m8e8err.h

```
_M8E8LastError@4
long M8E8LastError(long modno)
long modno,              // module number
```

D.4 m8a4.dll

- **M8A4Open** - The function returns 0 if the operation was successful, otherwise -1 is returned.

```
_M8A4Open@4
long M8A4Open(long modno)
long modno,        // module number
```

- **M8A4Close** - The function returns 0 if the operation was successful, otherwise -1 is returned.

```
_M8A4Close@4
long M8A4Close(long modno)
long modno,        // module number
```

- **M8A4WriteChannel** - writes a value to a analog output channel

```
_M8A4WriteChannel@12
long M8A4WriteChannel(long modno, long channel_no, long value)
long modno,              // module number
long channel_no,         // channel number
long value               // value
```

- **M8A4SetRelay** - connects D/A outputs to connectors (onOff => 0 switches the relay off, 1 switches the relay on)

```
_M8A4SetRelay@8
long M8A4SetRelay(long modno, int onOff)
long modno,              // module number
long onOff               // relay control
```

- **M8A4LastError** - List of error codes in m8a4err.h

_M8A4LastError@4
long M8A4LastError(long modno)
long modno, // module number

D.5 mbe20.dll

- **be20_open** - returns file descriptor if successful

_be20_open@4
long be20_open(long modno)
long modno, // module number
if return value is negative an error has occurred:
-1 - CreateFile() failed
-2 - ioctl to open new device failed
-3 - module driver can't connect to carrier driver
-4 - device limit exceeded
-5 - can't read registry

- **be20_close**

_be20_close@4
long be20_close(long fd)
long fd, //file descriptor
returns :
0 - Zero is returned if the connection has been closed successfully.
- 1 - no module has been opened
- 2 - ioctl failure

- **be20_read** - returns input value

_be20_read@4
long be20_read(long fd)
long fd, //file descriptor

D.6 mba20.dll

- **ba20_open** - returns file descriptor if successful

_ba20_open@4
long ba20_open(long modno)
long modno, // module number
if return value is negative an error has occurred:
- 1 - CreateFile() failed
- 2 - ioctl to open new device failed
- 3 - module driver can't connect to carrier driver
- 4 - device limit exceeded
- 5 - can't read from registry

- **ba20_close**

_ba20_close@4
long be20_close(long fd)
long fd, //file descriptor
returns : 0 - Zero is returned if the connection has been closed successfully.
- 1 - no module has been opened
- 2 - ioctl failure

- **ba20_write**

_ba20_write@8
long ba20_write(long fd, long val)
long fd, //file descriptor
long val //set output to val

The functions returns zero if the operation has been finished successful. Otherwise one of the following error codes is returned:
- 1 - ioctl failure
- 2 - illegal module number
- 3 - illegal bit number
- 4 - module not open

- **ba20_set_bit**

_ba20_set_bit@8
long ba20_set_bit(long fd, long bit_no)
long fd, //file descriptor
long bit_no //activate bit indicated by bit_no

The functions returns zero if the operation has been finished successful. Otherwise one of the following error codes is returned:
- 1 - ioctl failure
- 2 - illegal module number
- 3 - illegal bit number
- 4 - module not open

- **ba20_clear_bit**

_ba20_clear_bit@8
long ba20_clear_bit(long fd, long bit_no)
long fd, //file descriptor
long bit_no //clear bit indicated by bit_no

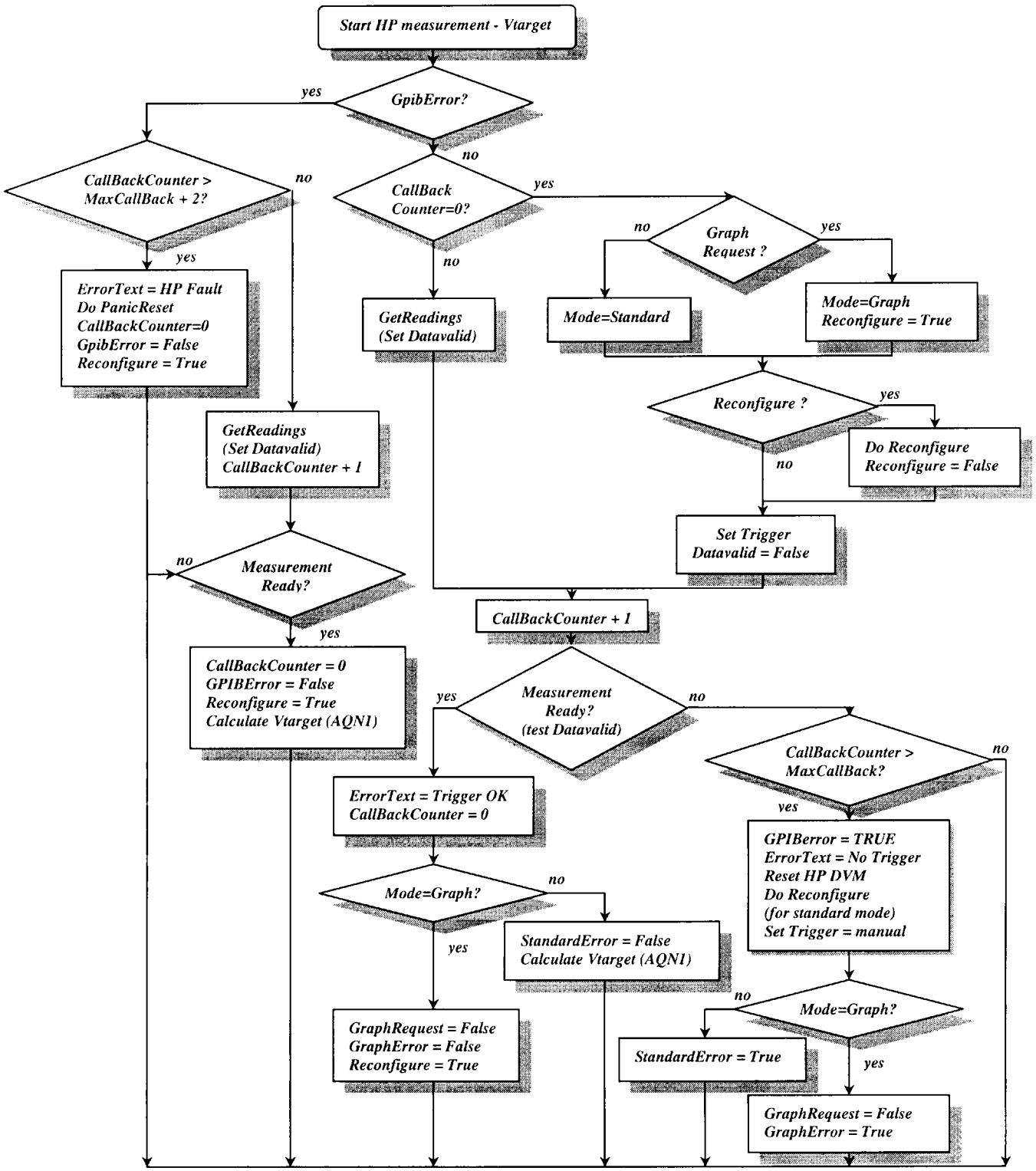
The functions returns zero if the operation has been finished successful. Otherwise one of the following error codes is returned.
- 1 - ioctl failure
- 2 - illegal module number
- 3 - illegal bit number
- 4 - module not open

- **ba20_get_shadow** - returns actual state of the output register

_ba20_get_shadow@4
long ba20_get_shadow(long fd)
long fd, //file descriptor

The functions returns zero if the operation has been finished successful. Otherwise one of the following error codes is returned.
- 1 - ioctl failure
- 2 - illegal module number

Appendix E - Flowchart HP-DVM measurement routine.



Distribution List:
PS-PO kicker section:
B. Bleus
S. Deman
H. Gaudillet
A. Fowler
J.C Freze
J. Schipper
L. Sermeus

PS-PO Group Leader
K.D. Metzmacher