CE
2002

PLM-5
PULSED PLATINUM NMR THERMOMETER
IEEE-488 (GPIB) Computer Interface

Remote Control Manual 1R4

PICOWATT
Veromiehentie 14
FIN-01510 VANTAA, Finland
phone 358 9 822087
fax 358 9 822184
Internet: www.picowatt.fi

Back to Links Index

QUICK COMMAND REFERENCE , FIRMWARE VERSION 1R4

Links Index for the PDF version of this manual - click the command headers

Back to Links Index

Operating states of the modules

Calibration parameters of the NMR module

Commands for calibrating the NMR measurement

Queries for outputs from the NMR module

Queries for outputs from the CS-10 module

Back to Links Index

Back to Links Index

INTRODUCTION

We call this interface "IEEE-488.2 aware". It is not in complete compliance with the standard, although effort has been made to achieve compliance in most cases. The design principles of the **PLM-5** and the principles of the **IEEE-488.2** standard differ in some respects. For example, in the **488.2** standard, the "state" of a device means the complete set of all its settings and values of programmable parameters. A "command" is then issued for starting the desired action. In the **PLM-5**, the settings and parameter values do not yet make a "state". Instead, a "state" includes also the operational state, where the **PLM-5** makes the desired action either only once or repeats it at previously specified intervals (or does not do anything). Because of this difference in philosophy, some deviations from the standard have been necessary. We have tried to point out all deliberate deviations. If the operation fails to comply with the standard in some important cases that we have not observed, please let us know.

States Instead of Commands

Setting values to measuring parameters, like gain or transmitter burst length, is typically not thought of as being a command, although this is made by giving a GPIB command message. On the other hand, making an NMR measurement sounds like a command. In fact, a command message is used for changing the operational state, but there is no command like "make one NMR measurement".

In the **PLM-5**, there is a variable called **MODE**. If this variable is set to zero, the NMR unit idles and no measurements are made. If **MODE** is set to 1, the **PLM-5** is allowed to make only one single NMR measurement, after which **MODE** is automatically returned to 0. If **MODE** is set to 2, the NMR unit is allowed to make measurements automatically at intervals determined by another variable called **Autointerval**. Similarly, the **CS-10** sweep can be set into a state where the reset is released so that the integrator's voltage increases or decreases at a predefined rate (**Sweep rate**) in order to follow and finally reach the target.

This idea of using ***states*** of operation instead of measurement ***commands*** brings some difficulty in deciding what **OperationComplete** means in this case. When is an ***automatic measurement*** complete? After each NMR measurement or after one week when the experiment is finished? Is a ***current sweep*** complete when the target has been reached, even though one hour may be needed for that? If other measurements are made during the sweep and these other measurements rely on the 488.2 defined **Operation Complete** command, confusion may arise when the completed sweep finally also reports about completion. What happens if the target is changed and ramping starts again without no other command? Our solution was to offer a possibility to query the ramp state from the status register and not to offer the ***OPC** command for current ramping at all.

Handling of the messages

All program messages to the **PLM-5** are handled sequentially, one after the other. As soon as all commands and queries on a message line have been processed, the new parameter values and the new operational states will come into effect.

According to 488.2, a (compound) program message line, whose maximum length is 255 characters, can be terminated by either the NL (newline, ASCII 10 decimal), ^END (end) or NL^END. These may also be called LF, EOI or LFEOI). In the **PLM-5**, the number of messages on a line is further ***limited to 20***. As soon as the **PLM-5** has received such a <PROGRAM MESSAGE TERMINATOR> (abbreviation = PMT), which must be the last item on a line, the input line is given to the parser, which converts all letters to uppercase and removes extra blanks from the string. If the original line contains several program messages, semicolons (<PROGRAM MESSAGE SEPARATORs>) between them, all such messages are separated and further divided into message headers and arguments. They are stored in two arrays, one for the headers and one for the arguments. If a line is continued after a PMT, unexpected failures may occur, and anyway the rest of the line will be neglected.

The parsed commands and/or queries from the array are given to the command interpreter. It compares each header in turn with the existing **PLM-5** command set. If a match is found, the corresponding data element is read or changed. A command can change a variable's value or it can change the state of the instrument. Any numeric data value is compared with its allowed range and if the value is acceptable, the corresponding **PLM-5** variable is replaced by the new value. Note that the changes do not come into effect immediately, if there are still more commands to be processed. A question mark ("**?**") as the argument (data element) indicates a query, and then the execution controller simply appends the current value of the requested variable to the output queue.

Once the complete input line has been processed, the command interpreter checks if the output queue is non-empty. If so, it is placed available for reading and the Message Available (MAV) bit in the status byte is set. The output consists only from responses to queries, the

Back to Links Index

PLM-5 does not add anything to the output by itself . The responses appear in the output in the same order as the corresponding queries in the input, and they are separated by semicolons. You have also the option of including the <RESPONSE MESSAGE HEADERS> in the output, which helps human reading, but usually makes machine reading more difficult.

About the Syntax

The <COMMAND PROGRAM HEADERS> AND <QUERY PROGRAM HEADERS> are insensitive to case. A misspelled or non-existing header results in a *Command Error*, if the argument is anything else than "**?**". If the argument is "**?**" and the header is misspelled, a *Query Error* will be reported. Such a command or query is bypassed and the **PLM-5** tries to execute the next command. One or more white spaces between the program/query message headers and the data items are allowed but not required. Note that if a parameter is left unchanged because of a command error, then the possibly following measurement may be made using wrong settings.

All <PROGRAM DATA> for the **PLM-5** is numeric, with the exception of the question mark ("**?**") that indicates a query. Non-numeric data results in a *Command Error* .

Depending on the data type required by the **PLM-5**, the value is then rounded and compared against the limits allowed for this specific parameter. An argument that is - after having been rounded- beyond its allowed limits as a **PLM-5** parameter but is otherwise valid for its type, results in an *Execution Error*. For example, the **PLM-5** may expect a byte variable that is between 0 and 19 but the data element in the command is 27. It is a valid byte but too large for this variable. A command with invalid data is ignored and the execution controller proceeds to the next command.

All data elements are first understood as real-number values, which allows for more flexibility in reception of numeric data. All numbers can be given either as (long) integers (e.g. 123), fixed-point decimal numbers (e.g. 123.45), or floating-point decimal numbers (e.g. 1.2345E+2). It is recommended, however, to use data in the same format as used by the **PLM-5** when it responds to the corresponding query.

Remote and Local

The "Remote Local" interface command has not been implemented, although it has been defined as mandatory in the 488.2 standard. The **PLM-5** front panel keys work in such a way that all changes in the variables are first written into the permanent ***Compact Flash*** memory, before making them effective. It is only then that the display is updated. This structure guarantees that the latest setup is always in good safe in case of any catastroph. This structure is not suitable for accepting remote commands in parallel with the local front panel control. That, in turn, is required by 488.2.

The solution was to disable most front panel controls during remote operation. The keyboard can be used for navigating within the top level and the highest **NMR** and **CS-10** levels, so that the graphic panes and all output text fields can be viewed. It is also possible to view the incoming and outgoing GPIB traffic for test purposes. No changes to the variables or instrument states can be made from the keyboard when the **PLM-5** is in the remote mode.

In the local mode, on the other hand, both the keyboard and remote commands can be used simultaneously without restrictions. It is on the user's responsibility to make sure that this possibility does not lead to unwanted results.

IEEE-488.1 requirements

SH1	Source handshake
AH1	Acceptor handshake
T6	Talker (no talk only mode)
L4	Listener (no listen only mode)
SR1	Service request complete capability
RL0	Remote Local. Mandatory but not implemented. A device-dependent REMOTE command controls remote and local operation and transition between these modes.
PP0	No parallel poll capability
DC1	Device clear complete capability w. selective device clear
DT0	No Device Trigger capability. Measurements can be started only by using device-specific commands
C0	No controller capability
E2	Electrical interface (75160 and 75161)

Back to Links Index

ABOUT THE PROGRAMMING EXAMPLES

General

This manual contains some programming examples. They have all been written in the old venerable **Turbo Pascal 6.0** language, but they should be quite understandable also for programmers using newer and more effective languages. Some examples are complete and working programs, while most are only one or two lines showing how to accomplish a task.

In order to write strings to and read strings from an **IEEE-488 controller card** in the PC computer, a driver is needed between the card hardware and the high-level program. In this case, our driver is a memory-resident "text file device". The programming examples use only few of its features and commands. The commands are not explained here, but their meanings have been commented in the examples. A Turbo Pascal unit **"IEEEIO"** was supplied by the manufacturer of the controller card, and it provides the interface between the TP program and the device driver of the card. All traffic between the **PLM-5** and the TP6.0 programs take place via two text files, which are opened in the IEEEIO unit.

• **IeeeOut**: Text file to which the commands are written as strings
• **IeeeIn**: Text file from which the response strings are read

As long as sufficient delays are used where needed in the program , operation of the driver is very simple and reliable. Also use of the **PLM-5** interface is reliable ***AS LONG AS THESE RULES ARE FOLLOWED:***

• Write to the **PLM-5** only when it is in a state where it can accept commands
• Read from the **PLM-5** only when there is something to read.

This sounds easy, but it is not always so. Please read the examples and the corresponding descriptions of the commands. Do not use delays for avoiding some programming lines. Instead, use serial poll loops.

Serial polling

With our driver, a device with address 22, for example, is serial polled by writing "SPOLL 22" to the controller:

```
writeln(IeeeOut,"SPOLL 22");
```

The **PLM-5** responds by sending one byte directly from its GPIB bus controller circuit. This operation is completely independent of what the **PLM-5** is doing, and therefore it can be serial polled at any time. The **PLM-5** can accept ordinary commands and respond to queries only when its state allows; this is because the program is sequential in nature and does not make use of interrupts. Serial polling, on the other hand, is allowed any time.

The response byte is received quickly, no delay should be necessary between serial polling and reading the response byte into a program variable:

```
readln(IeeeIn, Spr);
```

("Spr" stands for "Serial Poll Response").

Note that there may be some confusion about how the term "Serial Polling" is used. According to the IEEE-488.2 standard, serial poll returns a complete status byte, as was described earlier. In the **MatLab** software, for example, serial polling returns only the service request status of the polled instrument.

Therefore, **MatLab** may not be suitable for controlling the **PLM-5** in the intended way.

Service Requests

The serial poll response byte contains one special bit. This is bit 2^6 = 64. It is used to indicate that a device is "requesting for service". Such a request can be caused by an error that needs attention, such like reduction of gain. The device can also use a service request for telling that a task has been completed and that the computer can now read some results. All IEEE-488.2 compliant devices, including the **PLM-5**, have a complicated structure for enabling or disabling generation of a service request after a variety of ***events***. Some of those events are the same in all 488.2 compliant devices and some have been added by the manufacturer.

When a device generates a service request, two things happen: Bit 2^6 (the RQS message bit) in the serial poll response byte is set true AND a physical RQS signal line in the IEEE-488 bus is asserted. The signal line is wire-OR'ed so that any device in the same bus can assert the line and that it remains asserted as long as there is at least one device requesting for service. The controller card in the PC senses this line, and it can effect an interrupt. This provides a fast way to react to service requests, but it requires a free interrupt line and a lot of

Back to Links Index

programming skill and experience. We recommend that using interrupts is avoided.

If the service request line is asserted, the ***RQS Status*** of the controller card is true. With our driver, this can be verified by

```
writeln(IeeeOut, "SPOLL");
readln(IeeeIn, RqsStatus);
```

For this purpose, SPOLL must now be written without address to ***our*** GPIB driver. If at least one device is requesting for service, RqsStatus is 64 but otherwise it is 0. Your GPIB driver probably makes this differently. Please refer to its documentation.

When a device is serial polled, bit 2^6 in its response tells whether this instrument has requested for service, but it does not say anything about other instruments that are connected to the same bus. The RQS signal line, on the other hand, tells that at least one of the instruments has requested for service.

If you first detect the serial poll status of the controller, you will then need to poll each device separately in order to find the initiator(s). Then you may also need to inspect the various event registers in order to find out, what has caused the service request. It cannot be just anything, the reason must be one of those events that you have previously enabled to generate a service request.

Reading an instrument's status byte by serial polling **resets its RQS bit**, which will then remain zero until a *new reason* for service appears. Other parts of the status byte are not destroyed by reading it. If this is the only instrument in the IEEE-488 bus, also the RQS signal line is reset when this instrument is serial polled, but if there are more devices and if some other device is still requesting for service, the line remains asserted.

Using serial poll before writing commands

If you try to write to the **PLM-5** while it is busy, its GPIB card accepts only the first byte and then the bus hangs until the **PLM-5** is ready to process the input **OR** the bus timeout terminates the write operation. A spin-lattice NMR measurement may last for several minutes and the bus should by no means be hanged for all that time.

This possibility is eliminated by checking bit 2^7 of the status byte. It is set when the **PLM-5** is busy, and zero otherwise. A serial poll loop can be used for that.

```
PROCEDURE WaitUntilNotBusy;
{
waits until the busy flag, bit 2^7,
or the SPR becomes reset. Use before
trying to write to the GPIB interface.
Writing when the PLM-5 is busy may hang the
GPIB bus until its possible timeout.
Pressing any key terminates waiting and
program continues.
}
  var
    spr:byte;
    key: char;
begin
    repeat
      begin
        newdelay.delay(30);
        writeln(ieeeout,'spoll22');
        readln(ieeein,spr);
      end;
    until ((spr and 128) = 0) or keypressed;
    if keypressed then key:=readkey;
{
readkey is a TP function that returns the
character of the pressed key and resets
keypressed back to false
}

end;
```

This procedure serial polls device No. 22 continuously at 30 millisecond intervals until bit 2^7 in the response becomes zero or waiting is terminated by pressing a key on the computer's keyboard (useful when testing a program, later possibly a disaster. In real life, use a timeout). In a multitasking environment, the delay must be replaced by a delay procedure that does not use processor time. If the **PLM-5** is not busy, serial poll is performed only once.

Use this kind of serial poll loop routinely before every write operation in your program. Unfortunately, this method is not totally foolproof: In auto mode, the **PLM-5** can start a new measurement just after you have verified its state but before you have started to write commands. This possibility cannot be avoided by any other means than by using both modules only in the single modes.

Using serial poll before reading the response

One way to slow down the GPIB bus is to try to write to the **PLM-5** when its firmware is busy. The bus can be hanged totally by reading from the **PLM-5** when the output queue is empty. Therefore it is of utmost impor-

Back to Links Index

tance to check the output queue before trying to read. The following program example is a delay routine that waits until a message becomes available.

```
PROCEDURE WaitUntilMAV;
{
waits until the Message Available flag,
bit 2^4, of the SPR becomes set. Use before
trying to read from the GPIB interface.
Pressing any key terminates
waiting and program continues.
}
  var
    spr:byte;
    key:char;
begin
    repeat
      begin
       newdelay.delay(30);
       writeln(ieeeout,'spoll22');
       readln(ieeein,spr);
     end;
    until ((spr and 16) = 16) or keypressed;
    if keypressed then key:=readkey;
end;
```

This and the `WaitUntilNotBusy` procedure have been used when testing the GPIB firmware of the **PLM-5** and they have guaranteed reliable operation. In a multitasking environment, the delay should be such that it gives the processor time to other tasks. Its length is a compromise between released CPU time and speed of the waiting loop.

TIP: During the debugging phase you may inadvertently try to read an empty output buffer. In order to prevent the bus from hanging, issue the command GLBRESPALW1. Then the following happens: if the buffer is empty when reading from it, the string "ERROR 0" is placed into the buffer. The Message Available (MAV) flag is not affected, it remains false.

This way, the bus will not hang until timeout, but you may need to decode "ERROR 0" in your program. This is easy if you always read the response into a standard string variable instead of reading it directly into appropriate numeric variables. Although most responses are numeric, "ERROR 0" and response to the *IDN? query are strings.

STARTING TO CONTROL THE PLM-5 REMOTELY

Turning on the IEEE-488 Interface

The **PLM-5** may or may not have its IEEE-488 interface enabled. Go to ***Global Module/Interface/Type*** . Select IEEE-488 (The RS232 interface is not and will probably never be supported).

When started, the **PLM-5** will show some initialization details of the interface unit, if it is enabled. The message appears after the program has informed about the available memory:

```
Info: Gpib Addr:22  Status: ADMC=0 MODE=1 CLR=0
```

When the GPIB interface is enabled, the **PLM-5** can be controlled both remotely and locally. Note that remote control is possible always whereas local control is possible only when the remote mode is disabled.

Remote mode can be enabled/disabled only remotely.

The **PLM-5** starts always in the local mode.

You can, however, prevent remote control by disabling the interface. The state of the interface is saved and you need not enable it again for later sessions.

Changing the IEEE-488 Device Address

The default IEEE-488 device address of the **PLM-5** is 22. It does not have any secondary address (basic talker/listener).

Change the address from ***Global Module/Interface/GPIB address***. The address must be selected from range 1...31. The new address becomes valid immediately.

GPIB address can be changed only locally

Viewing the GPIB communications

It is useful to verify that commands are received and responses are sent properly before starting any serious work with the interface.

Back to Links Index

Go to the ***Global Module/Interface*** menu level. When you now send whatever characters or strings to the **PLM-5**, it shows them on the top message line.

Your GPIB controller software may include a simple program that can be used for communicating with an GPIB device. Using such a program, send some commands to the **PLM-5**.

send

```
OUTPUT22; *IDN?
```

the top line message should be

```
IeeeBus: Bus Input = *IDN ?
```

read the response to the above query

```
ENTER22
```

the top line should tell

```
IeeeBus: Bus Output= *IDN PICOWATT,
PLM-5,0,1R4
```

The GPIB viewer can also help in pointing out some errors.

Send the string:

```
OUTPUT22; *IDN?;ABC;*IDN?
```

This is a compound program message that consists of three semicolon-separated commands. The parser separates these three items. They are shown on the top message line one after the other, each for a few seconds. Then the execution control looks for the corresponding headers from the **PLM-5** command set. The "ABC" cannot be found, and the message line shows

```
IeeeBus: GPIB Command Error (ABC)
```

Showing commands and responses this way makes the **PLM-5** very, very slow. Use this facility only for testing and debugging purposes. Information about errors should be collected from the various event registers.

Back to Links Index

COMMAND REFERENCE

GLOBAL COMMANDS FOR GENERAL SETUP

GLBREMOTE[0..1 | ?] Remote Mode Command and Query
Data: Byte
Response: Byte

Use commands **GLBREMOTE1** and **GLBREMOTE0** for setting the **PLM-5** in the remote and local modes, respectively. Against to the 488.2 standard, we have omitted the ***Remote Local*** bus command, which turned out to be in contradiction with the design philosophy of the **PLM-5**. Instead, this device-dependent command for selecting the control mode is offered.

The **GLBREMOTE1** makes the following:
1) The "Remote" led lamp lights
2) The keyboard is inhibited so that no edits nor moving of cursors are possible. Only limited navigation is allowed for viewing the history panes and the output text fields
3) The state of the **PLM-5** is not affected otherwise. You can continue to operate the **PLM-5** right from the state it had when remote was selected. It is best, however, to enter the remote mode only from the idle state.

From the top level, you can descend only to the highest **NMR** and **CS-10** levels so that you see their history panes and output text fields, which are updated also in the remote mode.

You can also descend to ***Global Module/Interface***. When you are on this menu level, the top message line shows all command strings that the **PLM-5** receives from the IEEE-488 bus. A possibly long semicolon-separated command string is divided into separate commands with their arguments, and those commands are shown successively, each for a short time. The response, if queries were included, is then shown as a single line. If the command line contains errors, the ***last*** *command, query or execution error* is also shown. This feature can be a great debugging tool in the very beginning. Unfortunately, it makes remote control incredibly slow because of the delays for displaying data.

The graphic history displays, and the text output fields, on the **PLM-5** front panel must be configured manually before entering the remote mode. They do not belong to the remotely controllable items. However, they can be used to show the results that have been obtained under remote control.

When the **PLM-**5 enters the remote mode, the top level menu is automatically selected. Navigation is limited so that you can descend to the highest **NMR** and **CS-10** levels but not lower. So you can see all four history panes and all possible output text fields on the front panel also in the remote mode.

The screensaver works also in the remote mode. If no key has been pressed for 6 hours, the display is blanked. Press any key (ESC) to light the screen again.

The purpose of the Remote Mode is **not to enable** remote control, but to **disable** local commands and local changes to settings.

Use the command **GLBREMOTE0** to set the **PLM-5** in the ***local mode***. The following happens:

1) All variables including the states of the current supply and NMR modules are written on to the CF disk.
2) Data from the disk is read back, which changes the front panel display to comply with variables, modes and states that were in effect in the remote mode
3) The keyboard is released and the "remote" lamp is shut off.

The **PLM-5** continues its operation and, most notably, the magnet current remains unchanged. You should be able to continue manual operation right from the state where the remote computer left it.

The “Go Local” command GLBREMOTE0 can be given only from the GPIB controller. Is is not possible to disrupt remote control from the keyboard.

Over 60 files are written onto the CF disk after GLBREMOTE0. Therefore this command must be followed by a long delay, several seconds.

GLBTIME[0..23, 0..59, 0..59 | ?] Time Command and Query
Data: Comma separated hour, minute and second. Bytes.
Response: Comma separated hour, minute and second. All are two-digit bytes.

Back to Links Index

Set or ask the time of the **PLM-5**'s battery-backed clock. The command requires three parameters: hour 0..23, minute 0...59 and second 0..59, in this order and separated by commas. A query results the same items in the same order, also separated by commas. The parameters can be given as normal numbers like GLBTIME 7,8,9 or using two digits like GLBTIME 07,08,09 (eight minutes and nine seconds past seven in the morning). A query response uses always two digits. Out-of-range parameters result in an execution error and the time remains unchanged.

GLBDATE[1980..2099, 1..12, 1..31 | ?] Date Command and Query
Data: Comma separated year, month and day. Year is a 4-digit integer, month and day are bytes.
Response: Comma separated year, month and day. Year is a 4-digit integer. Month and day are two-digit bytes.

Set or ask the date of the real-time clock. The command requires three parameters: year 1980..2099, month 1..12 and day 1..31. Please note that the validity of the day with regard to months and leap years is not checked. The clock simply ignores day values of 29..31 if such a day does not exist for that month and for that year. Out-of-range parameters result in an execution error and the date remains unchanged. Parameters that are within ranges but do not exist in the calendar, like "September 31", do not result any error but the date may remain unchanged. You can compare the query response with the original command in order to verify that setting the date was successful.

GLBRESPALW[0..1| ?] Respond Always Command and Query
Data: Byte
Response: Byte
Default: Disabled

"1" Causes the **PLM-5** to respond with string "ERROR 0" if it is addressed to talk but the output queue is empty. ***This is against the IEEE-488 standard,*** but it may be useful when writing or debugging a new remote control program. If the device answers in any case, the IEEE-488 bus will not hang, waiting for timeout. Note that the MAV message output bit in the SPR Serial Poll Register remains reset. After you have debugged the control program, RESPALW should no longer be needed. But keeping it enabled should not do any harm, however.

GLBCLRPANES 1 Clear Panes Command (no query)
Data: Must be number 1
Response: no query

GLBCLRPANES1 clears the graphic panes (two for the NMR unit and two for the current supply) and resets plotting to re-start from the left edge, sample number 1.

All four panes are cleared so that they can be filled synchronously in auto modes, if data rates of both modules are equal. It is not possible to clear only some of them. You may want to use this command after having set the **PLM-5** in the remote control mode.

Note that whereas all collected data in memory is discarded, none of the 10 buffer files on the CF disk are cleared or deleted.

GLBHDRS[0..1 | ?] Include Response Message Headers Command and Query
Data: Byte
Response: Byte
Default: Disabled

With this command you can enable or disable the ***Response Message Headers*** from appearing in the response to queries. This can make the response to a compound query more "human readable" .

A compound query could be, for example, **CSDATARATE?;NMRAUTOITVL?**. If headers are included, the response is **CSDATARATE 5;NMRAUTOITVL 5**. If headers are disabled, the response is **5;5**.

Response headers contain only upper-case alphabetic letters from A to Z. The value may contain, in addition to digits 0..9 also the + and - signs, decimal point and letter **E** for the floating point exponent. There is always a white space between the header and and the value. Response messages to a compound query are separated by semicolons.

As the default, headers are disabled. Have them disabled also in your actual application program. Plain numbers are much easier to handle.

Back to Links Index

SELECTING ONE OF TWO SETUPS FOR THE NMR MODULE

NMRSETUP[0..1 | ?] Select NMR Setup Command and Query
Data: Byte
Response: Byte
Default at power-on: 0, the first or "old" setup

0 Use Setup 1 (“old setup”)
1 Use Setup 2 (“new setup”)

The selected setup applies for all subsequent settings of parameters and for all subsequent measurements until another setup is selected. Note that the keyboard works differently. There you have separate menu branches for setting parameters for both setups. Independent on where in the menu you are, measurements are made using the **NMR/Use Setup** selection.

NMRCOPYSETUP[1..2] Copy NMR Setup Command (no Query)
Data: Byte
Response: no query

Starts a copy operation *between* setups 1 and 2. Direction of the copy is determined by the argument: 1=copy setup 1 to setup 2, 2=copy setup 2 to setup 1. Any other argument results in an execution error.

NMRUSEINPUT[0..2 | ?] Selected Sensor Input Command and Query
Data: Byte
Response: Byte

Select one of the two input BNC connectors. 0=input 1, 1=input 2, 2=both inputs in parallel.

PARAMETERS FOR THE SELECTED SETUP OF THE NMR MODULE

NMRMODE[0..5 | ?] NMR Unit Operating Mode Command and Query
Data: Byte
Response: Byte

0 M0 (Curie mode),
1 T1 (Korringa mode),
2 Tune frequencies of the tank circuit and the filter (Tune Probe F mode),
3 Tipping Angle mode,
4 Test using an external signal (Test Ext Sig mode),
5 Test using an internal signal (Test Int sig mode).

Each of these measurements can set the OPC Operation Complete Bit both in single and automatic states. Also bit 2^6 (NMR measurement completed) in the NMREVENT register is set in these modes.

NMRNINETY[0..255 | ?] 90 Degree Burst Length Command and Query
Data: Byte
Response: Byte

The length is expressed in cycles of the NMR clock. Value 0 is required when measuring the tipping effectivity of the transmitter.

NMRTXMIT[0..255 | ?] Sampling Burst Length command and Query
Data: Byte
Response: Byte

The length is expressed in cycles of the NMR clock.

NMRTXAMPL[0..255 | ?] Transmitter Amplitude Command and Query
Data: Byte
Response: Byte

Calculate real peak-to-peak amplitude of the transmitter as 2*N*20 Volts /256. This is, however, not the exact voltage across the probe because it depends on the injection via two cross-coupled diodes and on the characteristics of the LC circuit.

Back to Links Index

NMRTTWODLY[0..255 | ?] T2 Delay Command and Query
Data: Byte
Response: Byte

The name means "NMR T2 Delay". It is the delay from the end of the 90 degree burst to the beginning of the first sampling transmitter burst in the T1 mode.

NMRTONEDLY[3..255 | ?] T1 Delay Command and Query
Data: Byte
Response: Byte

The name means "NMR T1 Delay". It is the delay from the end of the first sampling transmitter burst to the beginning of the second sampling burst in the T1 mode. Calculate the real exact delay time as (nmrtonedly-1)*0.09933+(96.5-nmrttwodly)*0.0010565 seconds.

NMRMODLY[0.1...30.0 | ?] M0 delay Command and Query
Data: Integer, fixed or floating point real
Response: Floating point real.

Delay between measurement of the static magnetization and the 90 degree burst in the T1 mode. *Note that the alphabetic header "NMRMODLY " contains the letter "O", not the digit "0" .*

NMRAUTOITVL[0..15 | ?]
NMR Autointerval Command and Query
Data: Byte
Response: Byte

Actual intervals for argument N are

N	interval		
0	1s	8	5 min
1	2 s	9	10 min
2	5 s	10	15 min
3	10 s	11	20 min
4	15 s	12	30 min
5	30 s	13	1 hour
6	60 s	14	2 hours
7	2 min	15	3 hours

NMRGAIN[0..15 | ?] NMR Gain Command and Query
Data: Byte
Response: Byte

Gain is defined as the DC input to the A/D converter divided by peak-to-peak signal at the probe. Approximate values when the filter is in wideband position for various arguments N:

N	Gain		
0	80	8	1500
1	110	9	2200
2	160	10	3100
3	230	11	4600
4	330	12	6500
5	490	13	9300
6	710	14	14000
7	1000	15	20000

NMRGAINRDGS[1..100 | ?] Number of Gain Stabilization Measurements Command and Query
Data: Byte
Response: Byte

Determines how many measurements are made using both the full and half stabilization amplitudes when stabilizing the gain. Averages are used in order to reduce noise introduced by the stabilization.

NMRGAINAMPL[0..255 | ?] Gain Stabilization Amplitude Command and Query
Data: Byte
Response: Byte

This is the amplitude that is generated by the transmitter inside the main unit. The signal is greatly attenuated in the preamplifier before it is applied to the input. Consider the values as relative only. Calculate the peak-to-peak transmitter output as 2*N*20 Volts/256. This is not a very useful value because of the high attenuation before injection to the probe.

PLM-5 units of various ages have different attenuation factors, we have not been able to decide what would be the optimum value.

Back to Links Index

NMRGAINSTAB[0..3 | ?] Gain Stabilization Method Command and Query
Data: Byte
Response: Byte

Specifies the method that the **PLM-5** uses for stabilizing the gain. The methods are:

0 stabilization off
1 Calculate both the gain and background. Subtract background and correct for any gain drift.
2 Calculate gain and correct for gain drift, but subtract an externally given background,
3 Only subtract an externally given background but do not alter the gain.

After a **NMRGAINSTAB[1..2]** command, the first gain stabilization measurement is used for fixing the *Gain Reference*, against which all subsequent gain stabilization measurements are compared. See the **PLM-5** manual for a detailed description.

NMRFREQRAN[0..5 | ?] NMR Frequency Range Command and Query
Data: Byte
Response: Byte

Ranges in kHz are:
0 15.6-**31.25**
1 31.25..**62.5**
2 62.5..**125**
3 125..**250**
4 250..**500**
5 500..**1000**

The numbers in boldface are used as the “names” of the ranges. These ranges are further divided in 256 steps, see below. The maximum frequency that can be synthesized on each range is 255/256 of the range.

NMRFREBYTE[125..255 | ?] NMR Frequency Command and Query
Data: Byte
Response: Byte

Together with the coarse frequency range, this frequency byte determines the synthesized NMR frequency in kHz as follows: F = N * range /256. For example, if N is 255 on the 250 kHz range, the frequency is 249.023 kHz. A small overlap of ranges was included so that N can start from 125 instead of 128. The low limit for N comes from the operating range of the simple synthesizer.

NMRFTUNEV[0..255 | ?] Bandpass Filter Tuning Voltage Command and Query
Data: Byte
Response: Byte

(The **PLM-5** documentation has used a variety of names for this parameter, like *Filtertune V, Varicap Volts, Ftune V*). It is the voltage across the filter’s capacitance diode (varicap). Calculate its actual value as 10 Volts * N /256.

> **NOTE:** In the wideband mode, firmware version 1R4 forces the filter tuning voltage to 255 (9.96 Volts). This is because the program sets this voltage to maximum in order to minimize loading of the signal. With older versions, you must set the voltage to 9.96 from the keyboard

NMRBANDW[0..1 | ?] Bandwidth Command and Query
Data: Byte
Response: Byte

0 wideband
1 filtered

The coarse range of the filter cannot be set remotely. It is necessary to open the preamplifier box and change the positions of some short-circuit pieces. Refer to the **PLM-5** operating manual.
See also the above note.

NMRSFID[0..4095 | ?] FID Integration Start Channel Command and Query
Data: Integer
Response: Integer

StartFID is the ADC channel from which the calculation of the surface area of the FID curve is started.

NMREFID[0..4095 | ?] FID Integration End Channel Command and Query
Data: Integer
Response: Integer

EndFID is the last ADC channel to take into calculation of the FID area. *EndFID* must be a larger number than Start FID. This is not checked by the program.

Back to Links Index

NMREXTBGND[0.0..4E+03 | ?] External NMR Background Command and Query
Data: Integer or fixed or floating point real
Response: Floating point real

If the gain stabilization procedure is not used, it is possible to subtract an externally given background from the magnetization readings. This not recommended, however. Refer to the **PLM-5** operating manual.

NMRNFWARN[0..1 | ?] Noise Floor Warning Command and Query
Data: Byte
Response: Byte

0 Disabled
1 Enabled

The warning is displayed on the **PLM's** status line also under remote control. The warning also controls a device-dependent error bit in the *NMREVENT* register. If desired, it can be enabled to generate a service request. The ADC overload bit and the noise floor warning bit together help the programmer to keep the FID signal at a suitable level.

NMRTHETA[0..90 | ?] Give External Tipping Angle Command and Query
Data: Byte
Response: Byte

If NMRTHETA is set to zero, the **PLM-5** determines the tipping angle of the sampling TX burst during each T1 measurement. This is the recommended setting, because it allows for flexible changing of the transmitter's effectivity.

If NMRTHETA is anything else than zero, the entered value is used when calculating temperature from the T1 time constant. Refer to the **PLM-5** manual for the temperature equation

NMRGREF[-1E5..+1E5 | ?] Gain Reference Command and Query
Data: Fixed or floating point real
Response: Floating point real

Do not edit this variable, except in the case described in the PLM-5 manual page 98. Value 0.000E+00 indicates that gain reference has not yet been set since gain stabilization was enabled.

PARAMETERS FOR THE SETUP OF THE CS-10 MODULE

CSTARGETA[0..50000 | ?]
CSTARGETB[0..50000 | ?] Commands and Queries for Ramp Targets 1 and 2
Data: Word (or long integer)
Response: Word

Arguments are positive regardless of the output polarity. Calculate output current as follows: I= N*10 A/50000 for the 10A range and I=N*2.5A/50000 for the 2.5A range.

CSRMPSPEED[0..7 | ?] Ramp Speed Command and Query
Data: Byte
Response: Byte

Speeds for the 10A range:		Speeds for the 2.5A range	
0	100uA/s	0	25uA/s
1	300uA/s	1	75uA/s
2	mA,	2	250uA/s
3	3mA/s	3	750 uA/s
4	10mA/s	4	2.5mA/s
5	30mA/s	5	7.5mA/s
6	100mA/s	6	25mA/s
7	1A/s	7	250 mA/s

CSOPRANGE[0..1 | ?] Output Current Range Command and Query
Data: Byte
Response: Byte

0 2.5A range
1 10A range

Note that a **remote** command will change the range immediately which will cause an abrupt change in the output current. This should usually be avoided in a thermometry application. Under manual front panel operation, change of output range is possible only when the output current is zero.

Back to Links Index

CSOPPOLAR[0..1 | ?] Current Output Polarity Command and Query
Data: Byte
Response: Byte

0 positive
1 negative

The current output is floating with respect to other parts of the **PLM-5**. "Positive" means that the red rear panel terminal is at a higher potential. Note that a remote command allows for immediate change in current polarity, which must usually be avoided in a thermometry application. In the local mode this has been prevented.

CSDATARATE[0..15 | ?] Current Supply Data Rate Command and Query
Data: Byte
Response: Byte

The name *"Data Rate"* refers to the **CS-10** module. It corresponds to the word *"Auto Interval"* of the **NMR** module. It is the rate at which both the output current and voltage are measured.

0=1 s	8=5 min
1=2 s	9=10 min
2=5 s	10=15 min
3=10 s	11=20 min
4=15 s,	12=30 min
5=30 s	13=1 hour
6=60 s	14=2 hours
7=2 min	15=3 hours

It is recommended to set the **CS-10** data rate equal to the Autointerval of the **NMR** module. Then the history panes of both modules are filled synchronously. On the other hand, information about a possible overvoltage is available only after a measurement of the output voltage, and it may be too late if the datarate is long. Then one has to use a short data rate and query measurement results directly from the RAM memory, not by reading the history buffer files. Even in that case, try to select such rates for both modules that they have a "common denominator". For example, do not select rates like 2 min for the **CS-10** and 5 min for the **NMR**, because then every second **NMR** measurement would not be accompanied by a current measurement. Intervals between the **NMR** measurements will then be slightly different. If **NMR** readings are taken so quickly that magnetization has not enough time to recover fully, e.g. when tuning the system, varying intervals are seen like noise in magnetization.

CSDATARATE is a variable that is used only in the "Run Auto" operating state.

CSMODE[0..1 | ?] Current Control Mode Command and Query
Data: Byte
Response: Byte

0=current controlled by the ramp
1=current controlled directly by the target value

The second mode enables fast slew rates. Only the first alternative (default) should be used in NMR thermometry where abrupt changes in current are not desired.

CSVOACTION[0..2 | ?] Overvoltage Action Command and Query
Data: Byte
Response: Byte

Determines the action taken if the output **voltage** of the current supply exceeds the safety limit. Alternatives are:

0 no action,
1 hold
2 shutdown

Hold means that the ramp is stopped, whereas shutdown means that the direction of the ramp is changed toward zero and a new speed, determined by CSSHUTDNSPD is applied. A fresh measurement result is required for deciding to start the specified *VoAction*. You need to use a fast data rate if you want to make best use of the safety features.

CSIOACTION[0..2 | ?] Overcurrent Action Command and Query
Data: Byte
Response: Byte

Determines the action taken if the output **current** of the current supply exceeds the safety limit. Alternatives are:

0 no action,
1 hold
2 shutdown

Hold means that the ramp is stopped, whereas shutdown means that the direction of the ramp is changed toward zero and a new speed, determined by CSSHUTDNSPD is applied. A fresh measurement result is required for

Back to Links Index

deciding to start the specified *IoAction*. You need to use a fast data rate if you want to make best use of the safety features.

CSVOLIMIT[0..10.0 | ?] Overvoltage Limit Command and Query
Data: Integer or fixed or floating point real
Response: Floating point real

Determines the safety limit for the output voltage in Volts. Regardless of the selected *VoAction*, exceeding this voltage causes a warning to be displayed on the **PLM-5** message line.

CSIOLIMIT[0..10.5 | ?] Overcurrent Limit Command and Query
Data: Integer or fixed or floating point real
Response: Floating point real

Determines the safety limit for the output current in Amperes. Regardless of the selected *IoAction*, exceeding this current causes a warning to be displayed on the **PLM-5** message line.

CSSHUTDNSPD[0..7 | ?] Shutdown Speed Command and Query
Data: Byte
Response: Byte

Determines the shutdown speed if a safety action is "shutdown" is selected to be a *safety action*. The shutdown speeds for the 10 A range are

0 100uA/s
1 300uA/s
2 1mA/s
3 mA/s
4 10mA/s
5 30mA/s
6 100mA /s
7 1A/s

Shutdown speeds for the 2.5A range are:

0 25uA/s
1 75uA/s
2 250uA/s
3 750 uA/s
4 2.5mA/s
5 7.5mA/s
6 25mA/s
7 250 mA/s

In the local mode, the **PLM-5** offers shutdown only at some of the above fixed pre-selected shutdown speeds. In the remote mode, you can realize also ***instant shutdown*** by ***just setting the ramp state to zero***. Should the NMR magnet quench, the current can be nulled quickly without need to worry about its heating effect. Note, however, that information telling about quenching is available only after a transaction with the **CS-10** unit. This calls for frequent measurements of the current and voltage so that the **CS-10** history panes fill quickly and do not provide much information about the past.

CSRELAYSW[0..1 | ?] Relay Switch Command and Query
Data: Byte
Response: Byte

Controls the reed relay switch, whose terminals are available on the rear panel of the **PLM-5** control unit.

0 open,
1 closed

The small current supply operates regardless of the relay switch position. See the **PLM-5** manual for a detailed description.

OPERATING STATES OF THE MODULES

NMROPSTATE[0..2 | ?] NMR Unit Operating State Command and query
Data: Byte
Response: Byte

0 Idle
1 Run Single
2 Run Auto

Refer to the **PLM-5** operating manual. Note, that in practice you can never get response "Run Single" for the NMROPSTATE? query, because the **PLM-5** is busy during a single measurement. And when it can respond, the state has already returned back to "Idle". In the auto state, on the other hand, there are idle times between measurements, and then you can get "Run Auto" as the response.

You can use the *OPC Operation Complete Command (or the *OPC?) query with both the single and automatic states. Enable the OPC bit by `*ESE1`. In single state, a command `*CLS;NMROPSTATE1;*OPC` would raise bit 2^5 (ESB) in the SPR as soon as then NMR

Back to Links Index

measurement has been made. The ESB can be further enabled to generate a service request. In the automatic mode, you must wait until the system is not busy and then issue `*CLS;*OPC`. The second automatic measurement would then set the operation complete flag.

See also the more specific NMRSTAT? and NMREVENT? queries.

CSOPSTATE[0..2 | ?] Current Supply Operating State Command and Query
Data: Byte
Response: Byte

0 Idle
1 Run Single
2 Run Auto

Operating state of the **CS-10** determines only how the output current and voltage are measured. The ***ramp state*** has its own command.

CS-10 Operating state can be que ried only when the system is not busy. This means that the CSOPSTATE? query can never return "1", because the system is busy during a single measurement. The auto mode is reflected in bit 2^0 of the SPR Serial Poll Response. This bit is not, however, unique because it is a logical OR with the operating state of the NMR unit. The Run Auto state can be determined by inspecting the CSSTAT register, but this must be done when the **PLM-5** is not busy.

CSRMPSTATE[0..4 | ?] Current Supply Ramp State Command and Query
Data: Byte
Resoponse: Byte

0 zero
1 hold
2 ramp to zero
3 ramp to target 1
4 ramp to target 2

The ramp state is considered a state of the instrument where the output current tries to follow the output of the ramp integrator. ***It is not an operation, and therefore the *OPC command cannot be used for checking when the target has been reached.***

The CSRMPSTATE? query returns the Ramp State, but it does not tell whether the selected target or zero current has been reached. You can use the SPR Serial Poll Response and the CSEVENT register for getting more information about what the output current is doing.

If the Ramp State is set to 0, the current output terminals are internally shorted so that the output resistance is well below 1 ohm.

CALIBRATION PARAMETERS OF THE NMR MODULE

NMRKORRK[0..1000.0 | ?] Korringa Constant Command and Query
Data: Integer, or fixed or floating point real
Reponse: Floating point real

Korringa Constant **K** is typically 29.8 for Platinum. Real number. You can change this number for calibrating the **T1** mode at a known temperature and magnetic field.

NMRCURIEC[0...1E+05 | ?] Curie Constant Command and Query
Data: Integer, or fixed or floating point real
Response: Floating point real

Curie Constant **C** . Enter this constant for calibrating the **M0** mode, or let one of the **PLM's** calibration modes to measure and calculate it.

COMMANDS FOR CALIBRATING THE NMR MEASUREMENT

NMRCAKT[0..100.0 | ?] Known Temperature for Calibration Type "A" Command and Query
Data: Integer, or fixed or floating point real
Response: Floating point real

Calibration type "A" means *Known Temperature, Known Magnetization*. Enter the temperature in millikelvins between 0 and 100 mK before starting the temperature calibration.

NMRCAKM[0..1E+07 | ?] Known Magnetization for Calibration Type "A" Command and Query
Data: Integer, or fixed or floating point real
Response: Floating point real

Enter the known static magnetization **M0** before starting the temperature calibration.

Back to Links Index

NMRCACALC1 Calculation of Calibration Type "A" Command (no Query)
Data: must be number 1
Response: no query

This effects calculation of a new Curie Constant “C”. The calculated new C is based on the NMRCAKT and NMRCAKM values and ***it applies only to the previously selected setup***.

The newly calculated value can be queried with the NMRCURIEC?.

Any argument other than “1” results in an execution error.

NMRCBKT[0..100.0 | ?] Known Temperature for Calibration type "B" Command and Query
Data: Integer, or fixed or floating point real
Response: Floating point real

Calibration type "B" means *Known Temperature, Measure Magnetization* . Enter the known temperature in millikelvins between 0 and 100 mK before starting calibration of type “B”.

NMRCBREPT[0..100 | ?] Number of Repeats for Calibration Type "B" Command and Query
Data: Byte
Response: Byte

This is the number of M0 measurements to be averaged for temperature calibration type “B” . Averaging reduces noise introduced by the calibration but it is useful only if the temperature remains stable.

NMRCBITVL[0..15 | ?] Measurement Interval Between Repeats in Cal. Type "B" Command and Query
Data: Byte
Response: Byte

Determines the time interval between M0 measurements in temperature calibration type “B”.

0	1s	8	5 min
1	2 s	9	10 min
2	5 s	10	15 min
3	10 s	11	20 min
4	15 s	12	30 min
5	30 s	13	1 hour
6	60 s	14	2 hours
7	2 min	15	3 hours

NMRCBACTION1 Start Calibration Type "B" Command (no Query)
Data: must be number 1
Response: no query

Starts temperature calibration type “B” procedure. This procedure can set the OPC (operation complete) bit in the *ESR event status register. Further, it sets the “Calibration has completed” bit in the NMREVENT register. The calculated new Curie Constant applies only to the previously selected setup.

A premature stopping of the calibration procedure is possible using the NMRSTOP command. It can be given at times ***between the M0 measurements, when the system is not busy***. All other commands during the calibration procedure are ignored. An interrupted calibration leaves the C unchanged.

Any argument other than “1” will generate an execution error.

NMRCCREPT[0..100 | ?] Number of Repeats for Cal. Type "C" Command and Query
Data: Byte
Response: Byte

Calibration type "C" means *"Measure both Temperature and Magnetization Using the T1 Method"*. This measurement can be repeated in order to get a less noisy average. Calibration measurements can be repeated even if there is moderate drift in temperature.

NMRCCITVL[0..15 | ?] Interval Between Repeats of Cal. Type "C" Command and Query
Data: Byte
Response: Byte

Determines the time interval between complete **T1** measurements in the temperature calibration type “C” procedure.

0	1s	8	5 min
1	2 s	9	10 min
2	5 s	10	15 min
3	10 s	11	20 min
4	15 s	12	30 min
5	30 s	13	1 hour
6	60 s	14	2 hours
7	2 min	15	3 hours

Back to Links Index

NMRCCACTION1 Start Calibration Type "C" Command (no query)
Data: must be number 1
Response: no query

Starts temperature calibration type “C” procedure which ends up with calculating a new Curie Constant for the previously selected setup . This procedure can set the OPC (operation complete) bit in the *ESR event status register. It also sets the “Calibration has completed” bit in the NMREVENT register.

A premature stopping of the calibration procedure is possible using the NMRSTOP command. It can be given at times ***between the T1 measurements, when the system is not busy***. All other commands during the calibration procedure are ignored. An interrupted calibration leaves the C unchanged.

Any argument other than “1” will generate an execution error.

NMRXFREPT[0..100 | ?] Number of Repeats for Xfer Cal Command and Query
Data: Byte
Response: Byte

Detarmines the number of *complete pairs* of M0 mode measurements using the old and new setups in the Transfer of Calibration procedure (Xfer Cal). The procedure transfers the existing Curie Mode calibration from setup 1 (the “old setup”) to setup 2 (the “new setup”).

Temperature in the cryostat must remain rather stable during transfer of calibration

NMRXFITVL[0..15 | ?] Interval Between Xfer Cal Measurements Command and Query
Data: Byte
Response: Byte

Determines the time interval between each M0 measurement in the “Transfer Calibration” (Xfer Cal) procedure.

		7	2 min
0	1s	8	5 min
1	2 s	9	10 min
2	5 s	10	15 min
3	10 s	11	20 min
4	15 s	12	30 min
5	30 s	13	1 hour
6	60 s	14	2 hours
		15	3 hours

Note that the whole procedure takes a time of 2*N*interval.

NMRXFACTION1 Start Transfer of Calibration Command (no Query)
Data: must be number 1
Response: no query

Starts the Transfer of Calibration procedure. This procedure can set the OPC Operation Complete flag in the *ESR event status register. Further, it sets the “Xfer of Calibration Completed” bit in the NMREVENT register. In the end, Curie Constant of Setup 2 (the “new setup”) is changed. Curie constant -and therefore the calibration- of the "old setup" remains intact.

A premature stopping of the calibration procedure is possible using the NMRSTOP command. It can be given at times ***between measurements, when the system is not busy***. All other commands during the Xfer Calibration procedure are ignored. An interrupted calibration leaves the Curie Constant of of the "new setup" unchanged.

Arguments other than “1” will generate an execution error.

NMRSTOP Premature Stopping of Calibration or Transfer of Calibration Command
Data: no data
Response: no query

The NMRSTOP command corresponds to stopping calibration or transfer of calibration by pressing ESC in local operation. It can be given ***during these two procedures when the system is not busy***. If it used otherwise, an execution error will result. NMRSTOP has no argument.

Back to Links Index

QUERIES FOR THE OUTPUTS FROM THE NMR MODULE

NMRTCURIE? Curie Temperature Query
Response: Floating point real

Value of T_{curie} in millikelvins from the last T1 or M0 (if calibrated) measurement. Calculated by dividing the Curie Constant C by the static magnetization M0.

NMRTKORR? Korringa Temperature Query
Response: Floating point real

Value of T_{Korr} in millikelvins from the last T1 measurement. Calculated by dividing the Korringa Constant K by the spin-lattice relaxation time constant T1.

NMRMAGNA? M0 Query
Response: Floating point real

Static magnetization M0 ("NMR Magnetization A") from the last M0 or T1 measurement. Available in all modes. If gain stabilization has been enabled in the Curie or Korringa modes, M0 is the stabilized, not the “raw” reading.

The number describes the surface area of the FID waveform between the StartFID and EndFID integration limits divided by the number of summed ADC readings (or channels). In the Test Ext Sig and Test Int Sig modes, the number is proportional to the surface area of a “good” part of the signal.

NMRMAGNB? M1 Query
Response: Floating point real

Magnetization M1 ("MAGN B")from the last T1 measurement. Sample that is taken shortly after the “90 degree burst”. Meaningful only in the T1 measurement mode. Stabilized if gain stabilization has been enabled. Calculated from the FID as described above.

NMRMAGNC? M2 Query
Response: Floating point real

Magnetization M2 ("MAGN C") from the last T1 measurement. Sample that is taken typically in the middle of the recovery. Meaningful only in the T1 measurement mode. Calculated as described above.

NRMTCONE? T1 Query
Response: Floating point real

Spin-Lattice Relaxation Time Constant T1 ("Time Constant One") in ***seconds*** from the last Korringa Mode measurement. Available only in the T1 measurement mode.

NMRTCTWO? T2 Query
Response: Floating point real

Spin-Spin Relaxation Time Constant T2 ("Time Constant Two") in ***microseconds***. Available only in the M0 and T1 modes, otherwise= -1. The value is based only on the StartFID and EndFID points of the LogFID line. Therefore it has no reliable accuracy.

NMRGRDG? Gain Stabilization Reading Query
Response: Floating point real

Gain Reading in the M0 and T1 modes. This number represents the surface area of the good part of the *full-amplitude gain calibration signal*. It has been scaled so that if the A/D converter has saturated, which occurs at +/- 2.5 Volt signal level, the reading is 2500. In other words, this reading tells the level of the full-amplitude calibration signal in millivolts.

NMRBKG? Calculated Background Query
Response: Floating point real

Value is calculated by the gain stabilization procedure in M0 and T1 modes. Unless *External Background* is given, the gain stabilization procedure subtracts the calculated background from all magnetization readings before correcting them by the calculated gain drift.

NMRGDRFT? Gain Drift Query
Response: Floating point real

Value is calculated by the gain stabilization procedure. It is the ratio of
Last NMRGRDG / Initial NMRGRDG
The initial NMRGRDG was obtained from the first measurement just after the gain stabilization was enabled. A value of -1 indicates that gain stabilization is not enabled and working.

Remember to fix the gain reference by the NMRGAINSTAB[1..2] command at a suitable moment.

Back to Links Index

The ***calibration procedure*** fixes the gain reference automatically if stabilization is enabled. ***Transfer of calibration*** fixes it automatically if stabilization is enabled ***for the new setup***.

NMRTIPANGLE? Calculated Tipping Angle Query
Response: Byte

PLM-5 calculates the effectivity of the sampling transmitter burst in terms of the “tipping angle” in the ***T1 mode***. This happens only if NMRTHETA is set to zero. Otherwise, the given "Theta" is used for calculating the Korringa mode temperature.

You can also determine the effectivity of any burst length and amplitude (NMRTXAMPL) in the Tipping Angle mode (NMR mode 3). Set NMRTHETA ("Tip Angle" softkey) to zero, select a suitable sampling burst length using NMRTXMIT, make NMRNINETY to zero and make a single measurement. Its result will be used as reference.

Increase NMRNINETY to the value whose effectivity you want to know and make a new single NMROPSTATE1 measurement. Then ask the tipping angle by NMRTIPANGLE?. The value is 255 until measurements using both a 0-degree and non-zero preliminary bursts have been made so that calculation of Theta is possible.

NMRMAX? Maximum FID Amplitude Query
Response: Floating point real

This is the maximum amplitude of the FID signal in volts. The maximum reading is 2.5 which indicates that the signal has saturated. Exceeding this limit results in an OVL error in the NMREVENT register.

Overload that occurs well before StartFID is not very dangerous.

Show the FID on a graphic display pane so that you can estimate, how bad the orveload is and whether adjustments can still be postponed.

NMRLASTADC? Last NMR Measurement Time Query
Response: Six comma-delimited fields

Returns the date and time when the last NMR measurement was made. The response has six fields, separated by commas (year, month, day, hour, minute, scond). For example NMRLASTADC 2003,06,11,16,05,17 . Leading zeros are added if needed to make two-digit fields.

QUERIES FOR OUTPUTS FROM THE CS-10 MODULE

CSCURRENT? Output Current Query
Response: Floating point real

Absolute value of the output current of the **CS-10** in amperes. Regardless of output polarity, this reading is always positive.

CSVOLTAGE? Output Voltage Query
Response: Floating point real

Absolute value of the output voltage of the **CS-10** in volts. Regardless of output polarity, this reading is always positive.

CSLASTADC? Last CS-10 Measurement Time Query
Response: Six comma-delimited fields each having at least two digits

Returns the date and time when the last **CS-10** current and voltage measurements were made. The response has six fields, separated by commas (year, month, day, hour, minute, second). For example CSLASTADC 2003,06,11,16,05,17 . Leading zeros are added if needed to make two-digit fields.

STATUS OF THE NMR MODULE

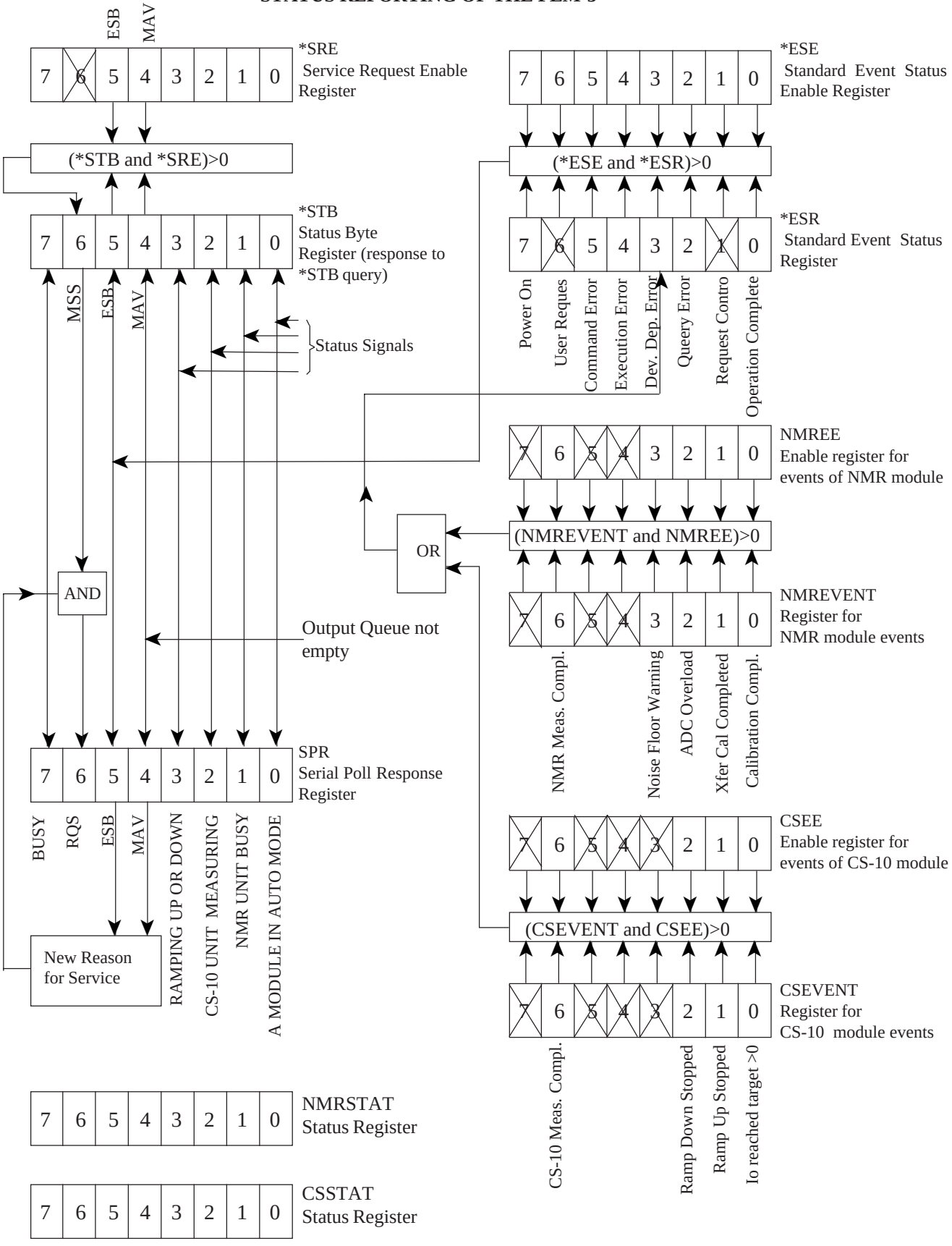
NMRSTAT? NMR Unit Status Query
Data: Only query
Response: Byte

Information that this status query returns is only whether the NMR unit is in auto mode or not. During all measurements and calibration procedures, the NMR unit is busy so that its status cannot be queried except by means of serial polling.

Bit	
7	In auto mode
6	0
5	0
4	0
3	0
2	0
1	0
0	0

Back to Links Index

STATUS REPORTING OF THE PLM-5



Back to Links Index

Even this single bit is useful, because bit 2^0 of the SPR Serial Poll Response tells only that "**either** the **CS-10**, the **NMR** unit **or both** is/are in auto mode".

NMREVENT? NMR Module Event Query
Data: only query
Response: Byte

This register contains five events. The *OPC command can be used instead of the NMREVENT information, but the latter is more specific. The device-specific events are handy when the current supply and the NMR module measure automatically, and especially if the measuring rates are different.

Bit	
7	0
6	NMR measurement was completed
5	0
4	0
3	Noise floor warning was issued
2	ADC overload was detected
1	Transfer of calibration procedure was completed
0	Calibration procedure was completed

This register is cleared when it is read. Also *CLS clears the NMREVENT register.

NMREE[0..255 | ?] NMR Module Event Enable Command and Query
Data: Byte
Response: Byte

Set the corresponding bits for enabling events specified in the NMREVENT byte. These are OR'ed with enabled events in the CSEVENT register and placed into bit 2^3 (DDE) of the *ESR event status byte. The DDE can further be enabled to generate a service request. See the picture about the status reporting structure on the previous page

STATUS OF THE CS-10 MODULE

CSSTAT? Current Supply Status Query
Data: only query
Response: Byte

Bits in this register are dynamically updated according to the phase of the firmware. The bits have the following meanings:

Bit	
7	In auto mode (0= idle, 1=auto)
6	0
5	In hold state
4	0
3	0
2	Ramping Down
1	Ramping Up
0	At nonzero target (target has been reached or supply is in hold state)

In order to find out the state of the ramp, the **PLM-5** firmware has to transact with the **CS-10** unit. Therefore the response is not immediate. Check with bit 7 of the SPR when the CSSTAT can be queried and check with MAV when the response is ready to be read.

Bit 7 is self-explanatory. Usually the controlling computer also knows if it has set the current supply unit in the auto mode, but if the state has been changed from the keyboard (possible only in the "local" mode), then this information may be valuable.

Bit 6 is for internal purposes only.

Bit 5: Normally, the controlling program knows when it has placed the **CS-10** in the hold state. However, hold state can also be a safety action, or someone have selected it from the keyboard in the local mode.

Bits 2 and 1: If the **CS-10** is ramping up or down, the output current will not be zero. If the current supply is **not** ramping either up or down, it may have **reached the target**, the current may have been **reduced to zero** or the supply may be **in the hold state**. Use bits 0 and 5 together with bits 1 and 2.

Bit 0 is one only when the **CS-10 has reached a nonzero target.** The bit is zero if the supply is still ramping (as shown by bits 1 and 2) or if the selected target is 0. Regardless of whether the current is zero or not, this bit is also asserted if the **CS-10** is in the hold state.

In the idle state, the CSSTAT is zero.

CSEVENT? Current Supply Event Query
Data: only query
Response: Byte

This byte collects some events from the current supply. With the aid of the corresponding mask, one or more of them may be enabled to contribute generating the ESB (Event Status) bit of the SPR status byte and further, optionally enabled to generate a service request. This byte is reset when it is read. The *CLS Clear Status command also resets this byte.

Back to Links Index

Bit	
7	0
6	Measurement was completed
5	0
4	0
3	0
2	Ramping down has stopped
1	Ramping up has stopped
0	Current has reached nonzero target

Note that also the *OPC command and the *OPC? query can be used to indicate that a measurement is complete. The OPC bit cannot be used for synchronization with the **CS-10** ramp events.

Bits 0-2 can be determined without measuring current and voltage, whereas bit 6 is set as soon as they have been measured, either in the single or automatic mode. By enabling bit 2^6 of CSEVENT to set the ESB Event Status Bit, you can synchronize the controller to the **CS-10** measurements.

The CSEVENT is cleared when it is read. Also *CLS clears this register.

CSEE[0..255 | ?] Current Supply Event Enable Command and Query
Data: Byte
Response: Byte

Enable any of the events in the CSEVENT byte by setting the corresponding bits in the CSEE register. The enabled bits are OR'ed and they are further OR'ed with the enabled bits from the NMREVENT register. The result is then placed into ESB bit 2^3 of the *ESR Event Status Register. ESB can be further enabled to generate a service request.

IEEE-488.2 DEFINED COMMON COMMANDS AND RESET COMMANDS

SPR Serial Poll Response and *STB? Status Byte Query

Refer to the diagram about the **PLM-5** status reporting. All information from the instrument is directed to the *STB and SPR registers with the exception of bit 2^6.

In case of the *STB register, bit 2^6 is called the MSS Master Status Bit. It is obtained from the ESB and MAV bits by first enabling any of them and finally ORing the result.

If STB bit 2^6 is zero, it is copied to bit 2^6 of the SPR register. Here this bit is called the RQS, Service Request Bit.

If STB bit 2^6 is one, it is copied to the RQS bit only if there is a new reason for service.

The RQS bit is cleared when the SPR register is read (by means of serial polling). The corresponding bit in the *STB register may remain set until the condition that has raised it has disappeared. The RQS bit is NOT cleared when the *STB register is read.

This arrangement guarantees that an IEEE-488.2 compliant device can request service only once per new reason.

Bits in the SPR and *STB registers have the following meanings:

Bit 7
One indicates that the non-multitasking **PLM-5** firmware is in a phase (=BUSY) where it **cannot** receive commands or queries from the remote controller. If a command or query is sent when bit 7=1, the **PLM-5** may hang the bus for a long time.

> Note that Bit 7 in the *STB status byte does not give any useful information about the instrument state, because *STB can be queried only when the system is not busy. Its value must be verified by serial polling.

Verify this bit always before issuing new commands (i.e. "if SPR<128 then...").

Even if bit 7 is checked before issuing new program messages, there is a very small possibility of hanging the bus temporarily **in** CS-10 or NMR **auto modes**. This can happen if the **PLM-5** starts a new automatic measurement after you have polled for bit 7 but before you give a new program message. In order to avoid this possibility, we recommend that only single mode measurements be used.

Bit 6
This bit behaves differently if it is read by serial polling or by using the *STB? query.

"1" indicates that the **PLM-5** has at least one **new** reason to request for service. Generation of the service request is one of the most hard-to-understand things in the IEEE-488.2 standard. At the highest level of all status reporting, a 488.2-compatible device has a Status Byte called *STB (the asterisk means a register that has been defined by the 488.2 whereas registers defined by the manufacturer do not have an asterisk). The *STB has a copy, which **we** call the **S**erial **P**oll **R**esponse SPR (it does

Back to Links Index

not have a good name in the 488.2 standard and other manufacturers may call it otherwise. Some vendors call both copies STB, which may be misleading).

A device's state is described by its **"status"**. Some aspects of this overall status are usually reflected by various status registers, which are on the device manufacturer's account. For example, one bit in the CSSTAT register indicates whether the **CS-10** module is in the auto mode. Another bits indicate ramping upwards and downwards. The 488.2 standard defines only one status bit in the *STB register, namely the MAV Message Available bit 2^4. It is set if there is something to read in the output queue. Bits 2^7, and the four lowest bits 0-3 in the *STB and SPR registers are free to be used by the manufacturer for what he likes.

Any change in status is called an **"event"**. For example, completion of a measurement is an event. Occurrence of an overload error is another event. The IEEE-488.2 defines the following events:

a) power on **PON**, b) command error **CME**, c) execution error **EXE**, d) device-dependent error **DDE**, e) query error **QYE** and f) operation complete **OPC**

These events are reflected by various bits in the *ESR Standard Event Status Register. The value for the DDE Device Dependent Error bit must be supplied by the instrument manufacturer (see the diagram on page 24).

The user of a 488.2-compatible system can determine, which events he lets to set the ESB Event Status Bit of the *STB register. This is made by "enabling" one or more of the events appearing in the *ESR register. Enabling means that one sets the corresponding bits in the *ESE Standard Event Status Enable Register. For example, setting bit 2^0 in the *ESE enables the Operation Complete event in the *ESR. Occurrence of **any** enabled event **is enough** to set the ESB, which is a logical OR of all enabled events.

The same structure of enabling various events is also applied to the DDE device-dependent error. The **PLM-5** has two module-specific event status registers, CSEVENT and NMREVENT. Any of those events can be enabled by registers CSEE and NMREE. Again, occurrence of any enabled device-specific event is sufficient to set the DDE bit in the *ESR register.

The **PLM-5** has also two "static" status registers, NMRSTAT and CSSTAT. They can be queried when the system is not busy, but status information as such cannot be used to generate a service request. Four status bits are also available as bits 0..3 of the SPR and *STB.

After the desired events have been enabled, the *STB status byte now contains the "static" MAV bit telling about the status of the output queue, and the dynamic ESB bit telling about events that have occurred. Either or both of these two bits can then be enabled, using the *SRE Service Request Enable Register, to set the **MSS Master Status Bit** in the *STB register.

Now comes the difference between the *STB and the SPR Serial Poll Response:

- If MSS is zero, the corresponding bit 2^6, called the **RQS bit**, in the SPR, is also zero regardless of what has happened in the device.
- *CLS sets RQS to zero.
- If MSS changes from zero to one, RQS changes to one
- If MSS is continuously one, but either the ESB or MAV changes from zero to one because of any reason, RQS changes to one. These two above situations are called "*new reasons to request for service*".
- If the SPR is read by means of serial polling, the RQS bit (but not the MSS bit) is reset. This quarantees that no device can ask for service more than once if there is no new reason.
- Bit 2^6 of the *STB remains set as long as the device has any reason, new or old, to request for service.

Whenever bit 2^6 of the SPR is 1, also a physical Service Request Signal line of the IEEE-488 bus is asserted. The state of this signal line can be used by the GPIB controller to effect an interrupt so that the remote control program can stop temporarily what it was doing, poll the devices for determining which one of them requested service, and then make the necessary actions before continuing its original task. Only skilled programmers should use interrupts. It is much easier to use serial polling and investigate the SPR byte as a whole or only its RQS bit.

It is the user's responsibility to make all necessary actions after a request for service has been detected. Such actions can include reading all event registers for checking, which events have contributed in the service request and which services are needed. After the services, the device status is usually cleared by *CLS, which makes the **PLM-5** ready to record new events.

If status is read by using the *STB? query, the MSS and RQS bits both remain intact. The MSS remains set as long as any enabled event exists whereas the RQS can be reset by serial polling. On the other hand, if MSS is zero, then RQS is also zero.

Serial polling can be done any time, whereas the *STB byte can be queried only when the **PLM-5 is not busy**.

Back to Links Index

Bit 5
This is the Event Status Bit ESB. It is the logical OR function of all enabled events. Refer to the diagram on page 24.

Bit 4
This is the Message Available Bit MAV. It is 1 if there is any output to be read (output queue is not empty).

Bits 0..3
Meanings of the four lowest bits have not been defined by the 488.2 standard. Therefore these bits are used for informing about some device-specific status in a form that can be accessed by serial polling and therefore independently of the phase of the sequentially operating firmware program.

Bit
3 Ramping current up or down
2 Current Supply unit is measuring
1 NMR unit is measuring, calibrating or transferring calibration
0 CS-10 and/or the NMR unit is in auto mode

Before issuing the *STB? query you must be sure that the **PLM-5** is ready to accept commands or queries. This is always made by serial polling. Therefore, it is questionable whether one should ever use the *STB command at all for reading the status, but only serial polling.

Read the note on page 7 regarding **Matlab** and serial polling

***SRE[0..255 | ?] Service Request Enable Command and Query**
Data: Byte
Response: Byte

This 8-bit register is used to determine, which bits of the *STB Register are enabled to generate a service request. In other words, which *STB bits are enabled to take part in changing the MSS Master Status Summary bit. The various *SRE bits have the following meanings:

7 Not used (internally set to 0)
6 Not used (internally set to 0)
5 Event Status Bit ESB.
4 Message Available Bit MAV.
3 Not used (internally set to 0)
2 Not used (internally set to 0)
1 Not used (internally set to 0)
0 Not used (internally set to 0)

So the only meaningful values are 0, 16, 32 and 48. The MSS Master Summary Bit in the *STB byte is one if (STB and SRE)>0 and zero otherwise. When updating the IEEE-488.2 status, the new *STB is compared with its previous copy. If the comparison shows that a new reason for service has appeared and if MSS is one, the RQS Request for Service bit in the serial poll response is set. For example, if you want the MAV to generate a service request, issue command *SRE16. Similarly, if you want to get a service request when an operation is complete, you must first enable the OPC bit in the Event Status Register *ESR (command = *ESE1). Then the OPC event will set the Event Summary Bit ESB in the status byte. Next, enable the ESB by issuing command *SRE32.
The *SRE register can be queried whenever bit 7 of the SPR is 0. Enable registers are not cleared by reading them.

Usually it is best to to read the SPR by serial polling instead of querying the *STB. The MAV bit is an example. Make a query, like NMRLASTADC?, then serial poll until bit 4 becomes true, and read the response. However, a serial poll loop needs CPU time and such a loop should not be used during a long procedure like transfer of calibration.

***ESR? Event Status Register Query**
Data: only query
Response: Byte

The *ESR register contains information about some events that are all specified in the 488.2 standard and also a summary bit that can be used to reflect the OR'ed events from the device.

The standard calls this bit **DDE**, **Device Dependent Error** Bit. We have widened the definition so that the DDE bit ***can include also other events than errors.*** This does not compromise the standard, as enabling such events depends on the user's decision.

The eight bits have the following meanings:

Back to Links Index

Bit

7 Power On (PON). This bit is set only once after the PLM-5 has been powered. Also the PONRESET command sets this bit

6 User Request (URQ). Not implemented.

5 Command Error (CME). A command error may result from a misspelled or unknown program message header. You can get more information by using the CMEERROR? query. It returns the offending command header without its argument. If more than one command message on a semicolon-separated line is invalid, only the last of them is returned by CMEERROR?

4 Execution Error (EXE). The argument following an otherwise valid command header is outside its allowed range for this command (for example, argument is 301 but maximum is 255), or the argument is otherwise invalid. You get more information about an execution error by using the EXEERROR? query. It returns the offending command header followed by the incorrect argument.

3 Device Dependent Error (DDE). See CSEVENT and NMREVENT for description about events that can be enabled to contribute this bit.

2 Query Error (QYE). Occurs when attempting to read from the **PLM-5** although the output queue is empty. Usually this happens when you send a misspelled query like "*IDM? " . Note that because of the misspelled query header, no response is placed in the output queue. If you try to read the response without first verifying that a message is available, the GPIB bus may hang until the controller's timeout.

Especially during debugging a new remote control program, the GPIB bus may hang several times because of attempts to read an empty output buffer. Neglecting the IEEE-488.2 standard, the **PLM-5** offers the **GLBRESPALW[0..1]** (Response Always) command. After RESPALW1, the **PLM-5** will respond with "ERROR 0" if no other output data exists. The MAV bit remains zero although the "ERROR 0" string is placed in the otherwise empty output queue. Preventing the bus from hanging can be useful if the bus timeout must be kept long for some reasons.

1 Request Control (RQC). Not implemented. The **PLM-5** cannot be a bus controller.

0 Operation Complete (OPC). This bit is affected by both single mode and automatic measurements, calibrations and transfer of calibration. It can also be used when synchronizing to the next automatic measurement.

The OPC bit is **not** affected by ramping the magnet current. Allowing ramping to control the OPC bit would make it impossible to use OPC together with any other measurement during the tens of minutes that might be needed by the ramp. Completion of the ramp is seen from bit 2^3 of the SPR serial poll response. More information about the ramp is available in the CSTAT and CSEVENT registers.

A query is not considered an operation and therefore queries do not set the OPC bit with the exception of the *TST? self-test query. After a query, check the MAV bit in the SPR before reading the response.

The *ESR register, like all event registers, can be read only destructively. This means that the *ESR? query resets the byte and it remains 0 until any of the specified events appears again. *ESR like all other event status registers is also reset by the *CLS command.

***ESE[0..255 | ?] Event Status Enable Command and Query**
Data: Byte
Response: Byte

Bits in this register enable or disable the corresponding events in the *ESR Event Status Register. The enabled bits are OR'ed for controlling the ESB Event Summary Bit in the *STB and SPR status bytes. For example, if you want both the Operation Complete and all errors to be shown in the ESB, set ESE equal to 2^5+2^4+2^3+2^2+2^0 or *ESE61 (*ESE= 32+16+8+4+1). Note that the more registers are enabled in the Event Status Bit, the less specific is the information that you can get.

***RST Reset Command**
Data: no data
Response: no query

The *RST command can be given only when the **PLM-5** is not busy. It resets the **PLM-5** by doing the following:

1 All parameter and variable values related to the NMR and current supply units are read from the CF disk, and these values replace the possibly different values that were in the firmware's volatile memory. This is in accordance with the IEEE-

Back to Links Index

488.2 standard, which requires that the device is brought to a state indepent of its past history.

2 Possible automatic modes are terminated and the OPC Operation Complete Bit in the STB Status Byte is set to zero. This is the only place where the *RST command changes the status registers of the device directly. Indirect changes in the status registers tell about the fact that the device has entered an overall idle state.

The *RST command does not

1 change the PLM's GPIB address
2 preserve the output queue (possibly in contradiction to the 488 standard). The queue is cleared.
3 affect the *ESE, NMREE or CSEE event status enable registers
4 affect the *ESR, NMREVENT or CSEVENT event registers
4 affect the power-on-status flag
5 affect any calibration data
6 preserve the history buffer, all buffers are cleared.
7 preserve any previous measurement data. All data is cleared
8 affect the *SRE service request enable register
9 affect the remote/local state

The *RST command should bring the device to a state that is independent of the past-use history of the device. However, changing the remote control state to local was thought unnecessary, because that is probably not the purpose of using *RST.

The *RST command resets the magnet current immediately, and all the settings of the **CS-10** and **NMR** modules are re-read from the CF memory.

The hard-wired measurement sequences of the NMR unit cannot be terminated remotely, and therefore one may need to wait until the **PLM-5** is ***is really idling before issuing the *RST*** command. Because the OPC bit is cleared, it cannot be used for telling to the remote controller that all the reset actions have been completed. This conflict is easiest to solve by using a sufficiently long delay after the reset command. ***About 20 seconds is needed.***

After that, verify by serial polling that the SPR Serial Poll Response byte returns zero. Typically, your next command will then be *CLS, Clear Status, for resetting all event bytes.

Use the *RST command with caution, because any possible output current from the **CS-10** is reset abruptly, which is seldom wanted in an NMR thermometry application. Use a sufficiently long delay (e.g. 20 seconds) after this command so that the **PLM-5** has time to complete all reset actions.

***CLS Clear Status Command**
Data: no data
Response: no query

This command clears all event status registers (*ESR, CSEVENT and NMREVENT) and sets the device into the operation complete idle state. It does not affect the actual measuring state of the **PLM-5**. For example, *CLS does not stop the current sweep nor reset the current and it does not terminate the auto modes of the **CS-10** or **NMR** units. An exception is the output queue. It does not belong to device status, and therefore it is not emptied by just clearing the status. In order to give correct information, also the MAV bit in the STB status byte must remain intact. This consideration is valid only if the *CLS is a part of a longer program message. If the *CLS is issued immediately after the previous program message, it will be handled as a new message and then it causes the output queue to be emptied. This will then affect also the MAV bit.

One way to use the *CLS command is the following:

1 Set all required values to the status enable masks (*ESE, CSEE and NMREE) if you want to direct any events or errors to the ESB Event Status Bit 2^5 of the SPR.

2 Set *SRE32 if you wish the ESB to further generate a service request. Set *SRE16 if you want the MAV bit to generate SRQ, or *SRE48 if you want both.

3 Put *CLS first on a new command line and after it, start the operation like CSOPSTATE1 (ask the CS-10 to make a single measurement of output current and voltage). Terminate this line with *OPC if you wish to get information about when the measurement is ready.

As program lines:

```
writeln(ieeeout,"*ESE1;*SRE32);
writeln(ieeeout,
  "*CLS;CSOPSTATE1;*OPC");
```

- serial poll until RQS is found, or detect the interrupt

Back to Links Index

```
WaitUntilNotBusy     {see page 8}
writeln(ieeeout,"CSCURRENT?");
WaitUntilMAV         {see page 9}
readln(ieeein, current);
```

Note that the waiting loops do not wait if the condition is true immediately.

DCL Device Clear and **SDC Selective Device Clear** are IEEE-488 bus commands. How to give them, depends on the 488 controller and its software.

The DCL/SDC bus commands can be **given** any time, but they have effect only after the sequential **PLM-5** program polls its GPIB routine in order to see a possibly changed status. Therefore, the response may not be instantaneous. The format of these commands depends on your GPIB card and its driver software. The effects of these commands are specified by the 488.2 standard. In the **PLM-5**, they

1 clear the output queue. The MAV bit in the STB status byte becomes zero.

2 set the **PLM-5** in the Operation Complete Command and Operation Complete Query idle states. The OPC bit in the STB status byte becomes zero.

3 clear the input buffer.

It is best to insert a sufficient delay after this command, set the instrument to idle states and to clear the status.

The **DCL/SDC** commands will not

1 change any programming data nor clear the history buffer

2 change the instruments measuring state in any way. This is important especially in the local mode, where one must be able to rely on the so far given front panel commands.

PONRESET Power-on Reset Command
Data: no data
Response: no query

The Power-on Reset Command is very similar to the *RST command. **In addition** to *RST it

1 Initializes the IEEE-488 interface

2 Resets all event status and event status enable registers

3 Sets the power-on bit (PON)

In other words, this command is equal to switching the **PLM-5** off and on again, exept that the modules are not scanned and no welcome screen is displayed.

A long time delay after this command is required.

***OPC Operation Complete Command**
Data: no data
Response: no query

The following operations in the **PLM-5** are able to set the OPC Operation Complete bit (bit 2^4) in the *STB status byte and SPR serial poll response:

*RCL
*SAV
*TST?
BUFSAVE
BUFRECALL
GLBCLRPANES
GLBREMOTE
all **NMR** measurements (see NMRMODE)
all **CS-10** measurements (see CSMODE)

Setting this bit is activated by first resetting the possible previously asserted OPC event in the *ESR event register by using the *CLS command. Then you give the actual command or set the instrument in the desired operational state (single or auto mode, calibration or transfer of calibration etc.). The command line is terminated by "*OPC". As soon as the operation has been completed, or a measurement has been made, bit 1 in the *ESR register is set to 1. If you have enabled this bit by asserting bit 1 of the *ESE enabling register, the OPC event will be reflected by bit 2^5 (ESB event status bit) of the SPR Serial Poll Response byte.

Back to Links Index

*OPC should be the last item of any command sequence, only then it can be used easily for synchronizing the **PLM-5** and the GPIB controller in the intended way. ***You should use the *OPC command together with only one such event that can set the OPC bit.*** For example, you might request both the CS-10 and the NMR unit to make one single mode measurement by using the following structure:

This is wrong:

```
WaitUntilNotBusy;
writeln(ieeeout, "*ESE1");
writeln(ieeeout, "*CLS;CSOPSTATE1;
NMROPSTATE1;*OPC");
- serial poll the SPR until bit 2^5 becomes
  1
- check that PLM-5 is not busy
writeln(ieeeout,"CSCURRENT?");
waitUntilMAV;
readln(ieeein, current);
writeln(ieeeout,"NMRTCURIE?");
WaitUntilMAV;
readln(ieeein, curietemp);
```

This will not work, because the CS-10 measures first and it sets the OPC bit before the NMR unit has even started its work. Therefore, you must use a longer method:

This is right:

```
WaitUntilNotBusy;
writeln(ieeeout, "*ESE1");
writeln(ieeeout,"*CLS;CSOPSTATE1;*OPC");
- serial poll the SPR until bit 2^5 = 1
WaitUntilNotBusy;
writeln(ieeeout,"CSCURRENT?");
waitUntilMAV;
readln(ieeein, current);
writeln(ieeeout,"*CLS;NMROPSTATE1;*OPC");
- serial poll the SPR until bit 2^5 = 1
WaitUntilNotBusy;
writeln(ieeeout,"NMRTCURIE?");
WaitUntilMAV;
readln(ieeein, curietemp);
```

Alternatively, you can make use of the device-specific event registers.

First, enable bit 2^6 of the NMREVENT register. This bit contributes in the ESB Event Status Bit by telling than an NMR measurement has completed. This is what you need:

```
WaitUntilNotBusy;
writeln(ieeeout,"NMREE64");
writeln(ieeeout,"*CLS;CSOPSTATE1;NMROPSTATE1");
-serial poll until bit 2^5 of the SPR
 becomes true
-read the responses:
WaitUntilNotBusy;
Writeln(ieeeout,"CSCURRENT?");
WaitUntilMAV;
Readln(ieeein,current);
Writeln(ieeeout,"NMRTCURIE?");
WaitUntilMAV;
Readln(ieeein,curietemp);
```

Note that now we have read both responses one after the other. This is possible because the NMR measurement is made last and only its completion is checked.

Also ***automatic measurements*** of both the **CS-10** and the **NMR** unit can set the OPC bit. Use this possibility for ***reading results from the next measurements.***

If you want to synchronize to the next **CS-10** measurement, set CSEE64;NMREE0 and if you are interested in the **NMR** unit, set CSEE0;NMREE64. As you then want to get the next results, just issue *CLS and wait until bit 2^5 of the SPR becomes true. Do not enable both modules at the same time. Then you may have difficulties in resolving, which one actually had set the ESB bit.

The **PLM-5** firmware is rather slow compared with the speeds of modern PC computers. The *CLS requires some time before it resets the ESB bit. If you let your PC program start serial polling immediately after the *CLS command, the first poll may show that ESB is still 1. If this happens, you must insert a delay in your program after the *CLS.

Other long-taking procedures can be followed with the help of the *OPC command, but there are also more specific event bits available in the NMREVENT register. Also the four lowest bits of the STB Status Byte can be used for verifying the **PLM-5** state using serial polling. The advantage of these registers is that the information is more specific than what the single OPC bit can give.

Although automatic modes can be used under remote control, we recommend that you use only the single mode measurements. Then synchronization between measurements and their results is quite easy to obtain.

You synchronize to the **next new measurements** by using the event registers and serial polling. **The last measurement results**, on the other hand, are always available to normal queries, except when the **PLM-5** is busy.

Remember to use byte 7 of the SPR Serial Poll Register for verifying that the **PLM-5** is not busy before writing anything to it.

Back to Links Index

***OPC? Operation Complete Query**
Data: no data
Response: Character "1" (or number 1)

This query places a charactor “1” (or number 1) in the output queue when all operations of the same command line have been completed.

It is used exactly in the same way as the *OPC command except that syncronization is now based on the Message Available MAV bit in the SPR serial poll response. We cannot see any reason for using the *OPC? query instead of the *OPC command. The OPC bit can be set only by completed operations whereas the MAV bit can be set by any other response as well. Therefore, the *OPC? query is less specific than the *OPC command and much less specific than the CSEVENT and NMREVENT status registers.

> If you have enabled response message headers by the command GLBHDRS1, you will not get a plain character “1” but the string “*OPC 1” which contains the query header. . Take this into account if you plan to use the *OPC? query for synchronization.

***WAI Wait-to-Continue Command (not implemented)**

This command has not been implemented in the **PLM-5**, although it has been defined as mandatory by the IEEE-488.2 standard. The sequential nature of the **PLM-5** firmware prevents *WAI from working as required.

The only situation where the *WAI command would be useful in **PLM-5** is to postpone all further actions until the current ramp has reached the target. Fortunately, it is possible to detect this by serially polling the SPR and by inspecting its bit 2^3. The bit remains set as long as the current supply is ramping either up or down. As soon as the ramp stops, the bit is reset.

All other time-taking commands, except the ramp, are able to set the OPC Operation Complete Flag and their completion-events are also available in the module-specific event registers.

Remember, that those registers can be read only if the **PLM-5** is not busy, which limits their usefulness. Only serial polling can be used always. The four lowest bits of the SPR together with the ESB and SQR bits should provide enough information for synchronization.

***IDN? Identification Query**
Data: no data
Response: string

The response consists of four fieds separated by commas:
1 PICOWATT (the manufacturer)
2 PLM-5 (model)
3 0 (zero, serial number is not available)
4 1R4 (or later firmware version)

***TST ? Self-Test Query**
Data: no data
Response: byte

The Self-Test Query can be made when the **PLM-5** is not busy. It is not very useful, but it has been defined as mandatory by the IEEE-488.2 standard.

*TST? addresses each of the three modules, namely the keyboard, the current supply and the NMR unit, in turn and checks their card version identification numbers. If these numbers match, self test returns 0. The possible errors are:

Bit
7 0
6 0
5 0
4 0
3 0
2 NMR unit does not respond or wrong ID code
1 CS-10 unit does not respond or wrong ID code
0 Keyboard unit does not respond

A successful test returns 0.

***SAV Save Command**
Data: no data
Response: no query

This command saves all parameter values and settings that are currently in the **PLM-5**’s volatile memory, onto the non-volatile Compact Flash disk. *SAV has ***no keyboard equivalence***, because all changes that are made in local mode from the keyboard are stored after each ENTER immediately on the CF disk.

When the **PLM-5** starts, all settings and parameters are read into memory from the CF disk files. ***Those files are kept up-to-date in the local mode***. In the remote mode, on the contrary, you make changes only to the values in the RAM memory. Those values are discarded if the **PLM-5** is re-started or if the disk files are read using the *RCL command.

Back to Links Index

The *SAV command is executed automatically when you enter the local mode by issuing REM0. This is necessary because the 488.2 standard requires that the Remote-Local transition shall not change the instrument's state and that the front panel indicators must agree with that state.

*SAV stores all settings of the **PLM-5**, ***including its operating states***. Be careful not to save states that you do not want to recall later. For example, if the recalled ramp state of the **CS-10** is zero, any possible magnet current will be reset immediately! In the opposite case, the current only starts to ramp toward the recalled target.

***RCL Recall Command**
Data: no data
Response: no query

The Recall Command reads all settings and operating states from the CF disk. Be very careful with this command, because if you have inappropriate settings on the disk, the current supply may ramp up to 10 amperes in some seconds.

The best use of this command may be together with interactive remote control where you may want to avoid long sequences needed for setting all parameter values.

If the **CS-10** ramp state is zero on the disk, recalling that setting will reset the magnet current immediately. Therefore, use *RCL with care!

QUERIES FOR ERROR DETAILS

CMEERROR? Command Error Query
EXEERROR? Execution Error Query
QYEERROR? Query Error Query
Data: no data
Response: string

Use these queries for getting more information about command, execution or query errors.

The error type, is found by inspecting the *ESR register. Once you know the type, you can then query more information about the error.

- The CMEERROR? returns only the command header, because only a faulty header can result in a command error.
- The EXEERROR? returns both the command header and the argument. An execution error can be caused only by an argument which is invalid, but it is necessary to know, which command was in question.
- A query error differs from a command error in that the argument has been a question mark "?". The QYEERROR? query returns both the faulty query header and the question mark.

These three queries are similar in that they return the header and argument of the last detected error.

For example, your command was

```
nmrninety35;nmrtxmit5;nmrgian6;nmrmode1
```

Bit 2^5 of the *ESR shows that there has been a command error. You can then query CMEERROR? and the response will be

```
NMRGIAN
```

If there are more than one error on a command line, and if they are of different types, the *ESR reports them all and you can inspect them all. If there are more than one error of one kind, only the last of them can be inspected.

COMMANDS FOR SAVING AND READING THE HISTORY BUFFERS

The **PLM-5** has ten sets of history buffer files. A set consists of one file for each graphic pane, four altogether. The panes are numbered 0 and 1 for the NMR unit, and 2 and 3 for the Current Supply Unit (higher and lower panes, respectively). A save operation (BUFSAVE) causes the oldest file set, number 9, to be deleted. Then files sets from 0 to 8 are renumbered from 1 to 9, and the currently visible data on all four panes is saved as file set number 0.

Under manual control, there is a keyboard menu command for reading one complete set of four files of a set into memory. First time that this command is used after power-on or after a save operation, set number 0 is read. Each time the keyboard command is given again, the file set number advances. After 9, the file set number is reset back to 0.

Under computer control, on the contrary, any single file from any file set can be read without need to go through them in sequence. You may want to read the buffers in an application where the **PLM-5** must remain

Back to Links Index

disconnected from the GPIB bus during a measurement. During such an application, the measurement data can be saved in successive buffer files manually or automatically once in an hour. You must first enable automatic saving from the keyboard (Top Display/All Panes/ Start Autosave).

> It is not possible to read the data direcly from memory via the GPIB bus. If you want to read the currently visible panes, you must first save them onto the CF disk using BUFSAVE.

BUFSAVE Save Buffers Command
Data: no data
Response: no query

BUFSAVE works exactly as its keyboard equivalent. The new data is stored as file set number 0, and the previous set 0 is renumbered as set 1 and so on.

BUFRECALL[0..9] Recall Buffer Command
Data: byte
Response: no query

BUFRECALL[n] loads the file set number **n** in the graphic display panes where it can be inspected using the cursor keys in the local mode. If no number is given, an execution error will result. BUFRECALLx cannot be used for reading the buffer files into a remote computer.

BUFSELECT[0..9,0..3 | ?] Buffer Select Command and Query
Data: Comma separated two bytes
Response: Comma separated two bytes

This command selects one of the ten buffer file sets and one of the four panes for **all** subsequent operations. The argument consists of two numbers separated by a comma.

The first number, ranging from 0 to 9, selects the file set. Number 0 means the most recently saved file set. The second number, ranging from 0 to 3, selects the graphic pane. Numbers 0 and 1 are for the upper and lower **NMR** history panes, and numbers 2 and 3 are the for the upper and lower **CS-10** history panes.

If the argument part is somehow invalid, the previous selection is left unchanged and an execution error is reported.

Response to the BUFSELECT? query tells which buffer file is currently selected. It consists of two digits, separated by a comma. Meanings of the numbers are described above.

BUFSTIME? Buffer Save Time Query
Data: no data
Response: six comma-delimited fields

This query is applied to the ***previously selected buffer file*** and it returns the time when the file was originally saved. Use it e.g. before deciding, which buffer file you want to read or which file set you want to recall.

Response is a string having the following comma-separated format: year, month, day, hour, minute, second. For example, BUFSELECT0,0;BUFSTIME? could return “2003,08,07,13,42,57”. Leading zeros have been added if necessary to make all fiels have at least two digits.

BUFFILL? Filling Degree of the Buffer Query
Data: no data
Response: integer

This query is applied to the ***previously selected buffer file*** and it returns the filling degree of it. A buffer can contain a maximum number of 359 records and each measurement adds a new record. As soon as all 359 places have been filled, 60 oldest records are discarded so that there is room for 60 new records. The filling degree in a “mature” buffer will therefore oscillate between 299 and 359.

Files within a file set will be filled equally ***only if*** both the **NMR** and **CS-10** units are in auto modes and if they make measurements at the same time (datarate = autointerval).

The buffer may contain empty records, which are calculated in the filling degree, however. This happens if something had prevented a module, operating in the auto mode, from making a scheduled measurement before it was time to do the next one.

You can use this query together with the BUFSTIME? query before deciding which one of the buffer files you want to read.

Back to Links Index

BUFRDSTATE[0..1] Buffer Read State Command
Data: byte
Response: no query

BUFRDSTATE1 sets the **PLM-5** in the buffer read state, where only one command and one query are allowed:

1 BUFREAD? query for reading one record

2 BUFRDSTATE[0..1 | ?] for exiting the buffer read state, for resetting the record counter to 0, and for asking the read state, respectively.

Buffer read state provides fast reading of the *previously selected buffer file* by means of preventing the **PLM-5** from doing anything else. The only things that you can do in this mode, are to read the records in their numerical order and to exit the state. Any scheduled automatic measurements of the **CS-10** and **NMR** units are interrupted as long as the device is in the buffer read state.

BUFRDSTATE0 exits the read mode immediately whereas BUFRDSTATE1 keeps the **PLM-5** in the read mode but resets the record counter so that reading re-starts from the beginning.

There is no query for asking the current record number. The data is intended to be read only sequentially and the first item of the response string is the record number.

BUFREAD? Read Next Buffer Record Query
Data: only query
Response: eight comma separated fields

Each new BUFREAD? query returns a response string with the following components, separated by commas:

1 record number 1..359 (integer)

2 value (usually a floating point number, integer in few cases)

3 date stamp (e.g. 2003,06,09 with preceding zeroes in month and day)

4 time stamp (e.g. 14,26,53 with preceding zeros, if required for two digits)

The query increments the record counter by one until record 359 is reached. If you try to read pass 359:

1 the MAV bit will ***not*** be set

2 the **PLM-5** exits the buffer read mode

3 the *ESR status register will show execution error

We recommend two methods to read a buffer. The first way is to read the complete file:

1 select the buffer using BUFSELECTx,y

2 enter the buffer read state by issuing BUFRDSTATE1

3 use the BUFREAD? query 359 times

4 exit the buffer read mode by issuing BUFRDSTATE0

The second way is to read only the existing records:

1 select the file using BUFSELECTx,y

2 query the filling state using BUFFILL? (=f)

3 enter the buffer read mode by issuing BUFRDSTATE1

4 repeat the BUFREAD? query f times

5 exit the buffer read mode by issuing BUFRDSTATE0

Remember always to verify -also in the buffer read state- by serial polling that output is available before trying to read it.

If a scheduled measurement in automatic mode has been skipped, the value field is blank. The buffer will almost always have some non-filled records between the last measurement result and number 359. The fields of these records contain only zeroes and empty strings.

Reading the buffer is limited to the data items that were selected to be plotted. For example, if the two NMR graphic panes show the Curie Temperature and the Korringa Temperature, you cannot read M0 from the buffer. Such items that are not plotted must be read from memory as soon as they are available, there is no buffer for them.

Back to Links Index

PROGRAM EXAMPLES

All four examples are working programs. They are written on Turbo Pascal 6.0 language and run under the DOS operating system.

```
Unit plmunit;


Interface
{
Interface section of a TP unit contains
 declarations of procs, functions, constants
 and variables that are available to the
 users of the unit
}
uses crt, dos, newdelay, ieeeio;

VAR
  spr: byte; {serial poll response}
  stb: byte; {status byte readable using a query}
  sre: byte; {service request enable register}
  esr: byte; {standard event status register}
  ese: byte; {std event status enable register}
  nmrevent: byte; {events of the nmr module}
  nmree: byte; {enable register for nmrevent}
  csevent: byte; {events of the current supply module}
  csee: byte; {enable register for csevent}
  nmrstat: byte; {status register for the nmr module}
  csstat: byte; {status register for the cs10 module}

  cmeerror: string; {command error, header}
  exeerror: string; {execution error, header and
argument}
  qyeerror: string; {query error, header and argument}


 PROCEDURE WaitUntilNotBusy;
 PROCEDURE WaitUntilMAV;
 PROCEDURE WaitUntilEsb;
 PROCEDURE ErrorDetails;
 FUNCTION Stderror:boolean;
 PROCEDURE GpibWrite22(outstring:string);
 FUNCTION GpibReadStr22:string;
 FUNCTION GpibReadItg22:integer;
 FUNCTION GpibReadReal22:real;




Implementation
{
The implementation section of a TP unit
contains the code of all procs and funcs
}


(*****************************************)
PROCEDURE WaitUntilNotBusy;
{
repeats serial poll loop until bit 2^7 of the
serial poll response becomes 0. This indicates
that the plm-5 is not busy and can accept
input from the gpib bus
}

var
  spr:byte;

begin
  repeat
    begin
      newdelay.delay(30);
      writeln(ieeeout,'spoll22');
      readln(ieeein,spr);
    end;
  until ((spr and 128) = 0) or keypressed;
{
keypressed provides a way to break an
infinite waiting loop
}
end;
(*****************************************)


(*****************************************)
PROCEDURE WaitUntilMAV;
{
repeats a serial poll loop until bit 2^4,
the Message Available Bit, becomes 1. This
indicates that the output buffer is not
empty and that it can be read
}
var
  spr:byte;

begin
  repeat
    begin
      newdelay.delay(30);
      writeln(ieeeout, 'spoll22');
      readln(ieeein, spr);
    end;
  until ((spr and 16) = 16) or keypressed;
{
keypressed provides a way to break an
infinite loop if the MAV never comes true
}
end;
(*****************************************)

(*****************************************)
PROCEDURE WaitUntilEsb;
{
repeats a serial poll loop until bit 2^5,
the Event Status Bit, becomes 1. The ESB
bit can result from many different events,
of which only one or few should be enabled
at a time. Otherwise ESB is too unspecific.
}
var
  spr:byte;

begin
  repeat
    begin
      newdelay.delay(30);
      writeln(ieeeout, 'spoll22');
      readln(ieeein, spr);
    end;
  until ((spr and 32) = 32) or keypressed;
{
keypressed provides a way to break an
infinite loop if the ESB  never comes true
}
end;
(*****************************************)


(*****************************************)
PROCEDURE ErrorDetails;
{using variables from the main program}
Begin
  if (esr and 32)=32 then
  begin
    WaitUntilNotBusy;
    writeln(ieeeout,'output22;CMEERROR ?');
    writeln(ieeeout,'enter22');
    readln(ieeein, cmeerror);
    writeln('Command Error: '+cmeerror);
  end;
```

Back to Links Index

```
  if (esr and 16)=16 then
  begin
    WaitUntilNotBusy;
    writeln(ieeeout,'output22;EXEERROR ?');
    writeln(ieeeout,'enter22');
    readln(ieeein, exeerror);
    writeln('Execution Error: '+exeerror);
  end;
  if (esr and 4)=4 then
  begin
    WaitUntilNotBusy;
    writeln(ieeeout,'output22;QYEERROR ?');
    writeln(ieeeout,'enter22');
    readln(ieeein, qyeerror);
    writeln('Query Error: '+qyeerror);
  end;
End;
(*****************************************)

(*****************************************)
FUNCTION Stderror:boolean;
begin
   WaitUntilNotBusy;
   writeln(ieeeout,'output22;*ESR ?');
   writeln(ieeeout,'enter22');
   readln(ieeein, esr);  {checking for cme, qye and exe
errors}
   if esr>0 then Stderror:=true else Stderror:=false;
end;
(*****************************************)

(*****************************************)
PROCEDURE GpibWrite22(outstring:string);
begin
  waituntilnotbusy;
  writeln(ieeeout,'output22;'+outstring);
End;
(*****************************************)

(*****************************************)
FUNCTION GpibReadStr22:string;
var response:string;
Begin
  WaitUntilMav;
  Writeln(ieeeout,'enter22');
  Readln(ieeein, response);
  gpibreadstr22:=response;
End;
(*****************************************)

(*****************************************)
FUNCTION GpibReadItg22:integer;
var response:integer;
Begin
  WaitUntilMav;
  Writeln(ieeeout,'enter22');
  Readln(ieeein, response);
  gpibreaditg22:=response;
End;
(*****************************************)

(*****************************************)
FUNCTION GpibReadReal22:real;
var response:real;
Begin
  WaitUntilMav;
  Writeln(ieeeout,'enter22');
  Readln(ieeein, response);
  gpibreadreal22:=response;
End;
(*****************************************)



BEGIN
{
This is the initialization section of a TP unit.
}

  spr:=0;
  stb:=0;
  sre:=0;
  esr:=0;
  ese:=0;
  nmrevent:=0;
  nmree:=0;
  csevent:=0;
  csee:=0;
  nmrstat:=0;
  csstat:=0;

  cmeerror:='';
  exeerror:='';
  qyeerror:='';

END.
```

Back to Links Index

```
program plm5_1;

{
In this example we send all necessary operating parameters to the
nmr and current supply units. The parameters were selected to
be suitable for a PRS-1 simulator. Possible command and execution
errors are checked. If no errors were reported, the new
parameters are saved onto the CF disk. They will be used in
another program examples about actual measurements.

This program uses the TP6.0 unit "plmunits" which declares
procedures WaitUntilNotBusy, WaituntilMAV and ErrorDetails.
Also the gpib register are declared there.

The program would be much faster if more than one command
were written on one line (<20 commands per line and a
total of 255 characters including separators)
}


uses crt, dos, newdelay, plmunit, ieeeio;

TYPE
  CommandArray=array[0..35] of string[15];

VAR
  gpibcmd, gpibarg: CommandArray;
  i:integer;


BEGIN    {main program}

{
disabling response message headers (disabled by default)
}
   GpibCmd[0]:='GLBHDRS'; GpibArg[0]:='0';

{
output range, polarity    and control mode
}
   GpibCmd[1]:='CSOPRANGE'; GpibArg[1]:='1';    {10A range}
   GpibCmd[2]:='CSOPPOLAR'; GpibArg[2]:='0';    {normal, pos. polarity}
   GpibCmd[3]:='CSMODE'; GpibArg[3]:='0';       {control via ramp}

{
writing the cs-10 parameters
argument for 2.62A: I=round(2.62 * 50000/10)=13100
}
   GpibCmd[4]:='CSTARGETA'; GpibArg[4]:='13100';     {2.62A}

{
other settings for the cs-10 modle
}
   GpibCmd[5]:='CSRMPSPEED'; GpibArg[5]:='5';      {ramp 30mA/s}
   GpibCmd[6]:='CSDATARATE'; GpibArg[6]:='5';      {measure at 30 s intervals}
   GpibCmd[7]:='CSVOLIMIT'; GpibArg[7]:='1.0';     {1V overvoltage limit}
   GpibCmd[8]:='CSVOACTION'; GpibArg[8]:='2';      {shutdown if overvoltage}
   GpibCmd[9]:='CSIOLIMIT'; GpibArg[9]:='3.0';     {3A overcurrent limit}
   GpibCmd[10]:='CSIOACTION';GpibArg[10]:='1';     {stop ramp if exceeds iolimit}
   GpibCmd[11]:='CSSHUTDNSPD';GpibArg[11]:='6';    {shutdown at 100mA/s}
```

Back to Links Index

```
{
parameters for the nmr module
}
   GpibCmd[12]:='NMRSETUP';GpibArg[12]:='0';       {use the first setup}
   GpibCmd[13]:='NMRNINETY';GpibArg[13]:='30';     {value OK for this prs-1}
   GpibCmd[14]:='NMRTXMIT';GpibArg[14]:='5';       {efficiency ca. 30 deg}

{
calculate argument for transmitter amplitude:
a=round(20.0Vpp * 256/40Vpp)=128
}
   GpibCmd[15]:='NMRTXAMPL';GpibArg[15]:='128';  {20Vpp at the transmitter}
   GpibCmd[16]:='NMRTTWODLY';GpibArg[16]:='50';   {50 if no better guess}

{
calculate t1 delay for 10 mK simulated
temperature, when T1 is about 3.0 seconds:
t1d=round(3.0/0.1)=30
Refer to "T1 delay command" for the exact
equation that must be used in calculation of
the Korringa temperature. T1 delay is required
only for the Korringa mode measurement.
}
   GpibCmd[17]:='NMRTONEDLY';GpibArg[17]:='30';     {t1delay 3 seconds}
   GpibCmd[18]:='NMRMODLY';GpibArg[18]:='1.0';      {m0 delay 1 second}
   GpibCmd[19]:='NMRAUTOITVL';GpibArg[19]:='5';     {auto interval 30 sec}
   GpibCmd[20]:='NMRGAIN';GpibArg[20]:='8';         {gain 1500 for this prs-1}
   GpibCmd[21]:='NMRGAINRDGS';GpibArg[21]:='5';     {average of 5 stab meas}

{
calculate argument required for a gain stabilization
amplitude of 10Vpp (from transmitter) as
a= 10*256/40 = 64
}

   GpibCmd[22]:='NMRGAINAMPL';GpibArg[22]:='64';       {g stab ampl 10Vpp}
   GpibCmd[23]:='NMRGAINSTAB';GpibArg[23]:='0';        {do not use}
   GpibCmd[24]:='NMRFREQRAN';GpibArg[24]:='3';         {125-250 kHz}

{
calculate argument required for 167.97 kHz
frequency:
f=round(167.97*256/250)=172
}
   GpibCmd[25]:='NMRFREQBYTE';GpibArg[25]:='172';         {frequency 167.97 kHz}

{
calculate argument required for 3.83V filter tuning
voltage
v=round(3.83*256/10)=98
}
   GpibCmd[26]:='NMRFTUNEV';GpibArg[26]:='98';        {3.83V across the varicap}
   GpibCmd[27]:='NMRBANDW';GpibArg[27]:='1';          {filtered}
   GpibCmd[28]:='NMRSFID';GpibArg[28]:='500';         {start from ch 500}
   GpibCmd[29]:='NMREFID';GpibArg[29]:='2000';        {end to ch 2000}
   GpibCmd[30]:='NMRNFWARN';GpibArg[30]:='1';         {enabled}
    GpibCmd[31]:='NMRUSEINPUT';GpibArg[31]:='0';      {normally use first input}
   GpibCmd[32]:='NMRMODE';GpibArg[32]:='0';           {Curie mode}
```

Back to Links Index

```
{
clear event registers so that possible errors can be recorded
}
  WaitUntilNotBusy;
  writeln(ieeeout,'output22;*CLS');

{
writing finally all the params
}
  for i:=0 to 32 do
  begin
    writeln(i,' '+GpibCmd[i]+' '+GpibArg[i]);
    GpibWrite22(GpibCmd[i]+GpibArg[i]);
  end;

{
check if there were any errors. If no, save params on CF disk
}
   GpibWrite22('*esr?');       {checking for command and exe errors}
   esr:=GpibReadItg22;
   if esr>0 then
     ErrorDetails
     else
     begin
       writeln('Parameters were sent - no errors');
       GpibWrite22('*sav');
     end;

END. {of the program}
```

Back to Links Index

```
program plm5_2;

{
After all parameters have been programmed, as in example
plm5_1, we start the current ramp from zero toward the
2.62A target. Output current and voltage, and also the
static magnetization are measured in the single mode during
ramping. When the target is reached, we check for adc
overload and noise floor errors. If either error occurs,
one has to find the reason. Then it may be
necessary to adjust e.g. gain, transmitter efficiency
or integration limits. This is best done manually after
having inspected the FID curve from the PLM-5 display.
   The PLM-5 will be calibrated in example plm5_3.
}

uses crt, dos, newdelay, plmunit, ieeeio;

VAR
  current:real;       {output current}
  voltage:real;       {output voltage}
  m0: real;           {static magnetization}

BEGIN
clrscr;

{
some initialization that is specific
to our driver
}
  writeln(ieeeout,'fill off');
  newdelay.delay(10);
  writeln(ieeeout,'TIME OUT 30');  {timeout in seconds}
  newdelay.delay(10);              {delay in milliseconds}


{
starting the ramp and curie mode
}
   gpibwrite22('*cls;csrmpstate3;nmrmode0');
   if StdError then
      ErrorDetails;
   gpibwrite22('glbclrpanes 1');    {clear panes}

{
prepare system so that operation complete sets the esb bit
in the spr register. see page 24.
}
   gpibwrite22('*ese 1');

{
following loop measures current, voltage and magnetization
in Curie mode at 10 second intervals  until the target is
reached. then program is terminated. Results are collected in
the history panes, provided that the cs10 panes are set for
current and voltage, and the nmr panes for M0 and FID.
}
   repeat
     begin
      {measure current and voltage}
```

Back to Links Index

```
       gpibwrite22('*cls;csopstate 1;*opc');
       WaitUntilEsb;
       gpibwrite22('cscurrent?');
       current:=gpibreadreal22;
       gpibwrite22('csvoltage?');
       voltage:=gpibreadreal22;

     {measure magnetization}
       gpibwrite22('*cls;nmropstate 1;*opc');
       WaitUntilEsb;
       gpibwrite22('nmrmagna?');
       m0:=gpibreadreal22;

     {write results on screen}
       writeln('Io=',current:8:5,' Vo=',voltage:8:5,' M0=', m0:8:5);

     {check if ramp has stopped}
       writeln(ieeeout,'spoll22');
       readln(ieeein,spr);
     end; {of the repeat loop}
   until (spr and 8)=0;    {not ramping up or down}

{
checking for ovl and noisefloor errors. Because *cls
has been repeated for each *opc, data in the nmrevent
register is valid and we can just look at it.
}
  gpibwrite22('nmrevent?');
  nmrevent:=gpibreaditg22;
  if (nmrevent and 4)=4 then writeln('Overload error');
  if (nmrevent and 8)=8 then writeln('Noise floor error');

IeeeComplete;   {command for our gpib driver}

END.
```

Back to Links Index

```
program plm5_3;

{
The PLM-5 has now been set to measure the PRS-1
simulator.
    In this program example, the PLM-5 is calibrated at a
simulated temperature of 10mK using the "Known Temperature,
Measure Magnetization" -method. Magnetization is measured
two times with enabled gain calibration.
    After calibration, the graphic panes are cleared and
measurements are made until a key on the PC keyboard is
pressed. The upper NMR pane should show Tcurie.
    The M0 pane is read from the PLM-5 in the last
program example plm5_4.
}

uses crt, dos, newdelay, plmunit, ieeeio;

VAR
  m0: real;           {static magnetization}
  curietemp:real;     {curie mode temperature}
  s:string;


BEGIN
clrscr;

{
some initialization that is specific
to our driver
}
  writeln(ieeeout,'fill off');
  newdelay.delay(10);
  writeln(ieeeout,'TIME OUT 30');  {timeout in seconds}
  newdelay.delay(10);              {delay in milliseconds}


{
writing params for gain stabilization and
calibration
}
   gpibwrite22('*cls');
   gpibwrite22('nmrgain 8');         {gain 1500}
   gpibwrite22('nmrgainampl 255');   {stab ampl 10Vpp at txmitter}
   gpibwrite22('nmrgainstab 1');      {full stabilization}
   gpibwrite22('nmrgainrdgs 3');      {gain is measured 3 times}

   gpibwrite22('nmrcbkt 10.0');     {calibrate at 10.0mK}
   gpibwrite22('nmrcbrept 1');      {m0 is measured twice}
   gpibwrite22('nmrcbitvl 5');      {interval 30 seconds}
   if StdError then
      ErrorDetails
      else
      begin
         gpibwrite22('nmree1;*ese 8');       {enable stop of calibr}
         gpibwrite22('*cls');                {into the esb bit}
         gpibwrite22('nmrcbaction 1');       {start calibration}
      end;
```

Back to Links Index

```
{
Before continuing after calibration, we must wait until
calibration ends.
}
   WaituntilEsb;
   if stderror then errordetails;

{
measure then the simulated temperature at 30 second
intervals. It is assumed that the upper graphic pane
has been set to show Tcurie. setting must be made manually.
Readings are taken when the ESB bit in the serial poll
response indicates that an nmr measurement has been completed.
In order to enable that, the corresponding bit must be enabled
}
   gpibwrite22('nmree 64');  {enable compl'ed nmr into dde}
   gpibwrite22('*ese 8');    {enable dde into esb}
   gpibwrite22('nmrautoitvl 5;glbclrpanes 1');
   gpibwrite22('nmrmode 0;nmropstate 2;*sav');  {turn on auto mode}

repeat
  begin
    gpibwrite22('*cls');
    waituntilesb;
    gpibwrite22('glbtime ?;nmrtcurie ?');
    s:=gpibreadstr22;
    writeln(s);
  end;
  until keypressed;

{
leave plm5 in idle state
}
  gpibwrite22('nmropstate 0');

{
save the graphic panes. this data is read
via the GPIB in the last program example
plm5_4.pas
}
  gpibwrite22('bufsave');


IeeeComplete;  {command for our gpib driver}

END.
```

Back to Links Index

```
program plm5_4;

{
In this last program example, one history pane is
read via the GPIB bus and saved onto the hard disk.
The selected file is the upmost nmr pane from the
last saved file set. All commands for selecting this
file are used although it is this file that is also
selected by default.
   The saved text file can be viewed with any text
editor.
}

uses crt, dos, newdelay, plmunit, ieeeio;

VAR
  i: integer;
  filldeg:integer;
  bufline: string;
  savefile: text;



BEGIN
clrscr;

{
some initialization that is specific
to our driver
}
  writeln(ieeeout,'fill off');
  newdelay.delay(10);
  writeln(ieeeout,'TIME OUT 30');  {timeout in seconds}
  newdelay.delay(10);              {delay in milliseconds}

{
Selecting the buffer file to be read and its
filling degree
}
  gpibwrite22('bufselect 0,0');  {latest set, first file}
  gpibwrite22('buffill ?');       {filling degree of selected file}
  filldeg:=gpibreaditg22;

{
open or create a text file in the current
directory for saving the buffer
}
  assign(savefile,'pane0fil.txt');
  rewrite(savefile);

{
  in the following loop the program enters the buffer
  read mode. Then a new line is read as a string. The
  string contains only commas as delimiters, and
  therefore it can be written directly onto disk.
  Some languages or applications may require that the
  commas are replaced by white spaces before writing
  onto disk.
     Note that procedure gpibwrite22 waits until the
  plm-5 is not busy, and function gpibreadstr22 waits
```

Back to Links Index

```
  until a message is available.
     As long as you do not read beyond 359, the response
  to bufread? will not be empty.
}
  gpibwrite22('bufrdstate 1');
  for i:=1 to filldeg do
    begin
      gpibwrite22('bufread ?');
      bufline:=gpibreadstr22;
      writeln(bufline);
    end;

{
remember to exit the buffer read state !!!
}
  gpibwrite22('bufrdstate 0');

  Close(savefile);

  IeeeComplete;       {command for our gpib driver}

END.
```

Back to Links Index

INDEX

Use page numbers as links in the PDF version of this manual

Symbols

A

B

C

Back to Links Index

Back to Links Index

Back to Links Index

O

P

Q

R

S

Back to Links Index

T

U

V

W

X