# Scripting of Nmr Sequences in AcqNmr

Ing.Dr.Stanislav Sykora, August 15, 2002

## I. Introduction

This Note describes the **pulser sequence programming language** employed by the STELAR AcqNmr data acquisition program (starting from **version 2.00**). The scripts employs a User- and application-oriented *fourth generation language* (4GL).

The definitions and descriptions which follow are comprehensive but very compact. The Author recommends that the interested reader complements reading this Note with a practical example provided in the Application Note S.Sykora, *Composite-pulse null-biased IR sequence*, of August 15. Alternatively, display the DefaultSequences.ssf file and use any of the listed sequences as an example.

## II. Sequence scripts and sequence files

A **pulser sequence script** consists of a contiguous segment of **plain, case insensitive ASCII text**. Its beginning is marked by a text line starting (apart wrom any white space) with the keyword **SEQUENCE**, followed by the name of the sequence. The sequence script terminates with a text line starting with the keyword **END SEQUENCE**. Individual pulser sequence segments thus appear as text blocks of the form

```
SEQUENCE SequenceName
....    'sequence-script body
....
....
END SEQUENCE
```

(attention: unlike all the rest of the script, the keywords **SEQUENCE** and **END SEQUENCE** MUST be in CAPITAL letters).

Any number of pulser sequences may be grouped together into a single **pulser sequences file**. The only additional requirement on such files is that, in order to be recognized by the software, their first line *must* start with the discriminator string

**STELAR Script File**

and, preferably, their file-type extension should be "ssf" standing for *"Stelar sequences file"*. Apart from the discriminator condition, there are no other limitations. Any text which does not appear within a **SEQUENCE**....**END SEQUENCE** text bracket is simply ignored - a fact which may be exploited conveniently for comments and/or descriptions of arbitrary extension.

Stelar supplies one pulser sequences file named **DefaultSequences.ssf** which should be located in the same directory as the executable file. The software, however, includes mechanisms for searching and selecting pulser sequences located in text files anywhere in the system (including a network). **Users are encouraged to write their own pulser sequences and store them in their own pulser sequence files.**

## II. Basic rules

There are some simple rules which hold everywhere within the body of a pulse sequence script:

**1.** Extra ***white space*** is ignored.
Things like strings of blanks and/or tabulators, for example, have the same meaning as a single blank. A new line is recognized by the LF (line-feed) character. CR (carriage return) is treated as a blank and so are all other ASCII control characters.

**2.** Apostrophe (') starts a ***comment*** which extends up to the end of the line.

**3.** ***Sequence header***.
Pure comment lines immediately following the SEQUENCE declaration line constitute the sequence header which is displayed in the preview box in experiment/sequence selection dialogs. It may (and should) contain a brief description of the sequence and instructions for the final User on how to use it.

**4.** ***Empty lines*** are ignored.
The only exception to this rule is header termination since an empty line is sufficient to terminate the macro header.

**5.** The script is structured as a series of ***instructions*** or ***commands***.
Longer commands are usually written one per line, in which case they do not require a special terminator (the **line-feed** is itself a valid terminator). Short commands, however, are often packed several per line in which case they must be terminated by **semicolons**.

## III. The two script Sections

The body of a pulser sequence script is divided into two sections: the **sequence-proper section**, followed by a **setup section**. The end of the sequence-proper section and start of the setup section is marked by the keyword **#SETUP**. The general structure of a pulser sequence script is therefore

```
SEQUENCE SequenceName
....
....    'sequence-proper section
....
#SETUP
....
....    'setup section
....
END SEQUENCE
```

The two sections are scanned by the software at quite different times and for quite different reasons. When a new experiment is loaded (parameter **EXP**), the syntax of the corresponding pulser sequence script is checked (both section). Should an error be detected, the script is rejected and the EXP parameter is not allowed to change. Otherwise, the whole script is loaded into a buffer and its *setup section* is scanned *in order to initialize the values and/or access flags and option strings of any specified parameters*.

The initial values and the option strings of the initialized parameters need not remain intact forever. If the parameter is User-accessible (or defined by means of a formula including other User-accessible parameters), its value may be changed at any time by the operator. At the moment of pulser programming, the actual, current values of each parameter must be used.

It is only *during actual pulser programming* that the *sequence-proper section* is scanned by the software *in order to generate the proper sequence of pulser steps*.

Notice that, while the setup section is scanned only once (just after changing the parameter EXP), the sequence-proper section is scanned every time the pulser needs re-programming because of a changed value of a pulser-related parameter. However, this does not imply re-opening of the source file since the sequence script is already stored in a computer RAM buffer.

## IV. The SETUP Section

The setup section of a pulser sequence script is scanned just once when loading a new experiment (parameter EXP). Normally the order in which its individual items appear is not important but should there be any interdependencies, you are invited to follow a logical way (just in case). Each of its commands consists of:

**1.** A parameter acronym, followed by the following *optional* items:

**2.** Parameter User-access configuration consisting of one of the following characters:

| | |
|---|---|
| = | enable User access |
| : | disable User access |
| space | don't modify |

**3.** Initial value to be set when the sequence is first loaded (remember that, subsequently, the value of the parameter may still change). A plain numeric value may be specified either directly as a number (any format) or as a string constant (e.g.,"23"). Values may be initialized also to expressions; in this case, they must be enclosed in parentheses and follow the *expression-value syntax rules* described in the Parameter Values chapter of the on-line manual.

String values must be "enclosed in quotation marks".

**4.** Initial settings of *pulser parameter options*. This optional item, if present, must be enclosed in square brackets. When the parameter is a *pulser interval*, the contents of the brackets are copied into its options field when the sequence is loaded (again, this does not preclude subsequent changes).

When any of the optional items is missing, the current value of the parameter is left unchanged.

In order to understand properly the initialization process, keep in mind that, upon loading a new pulser sequence but prior to any script-driven initialization, the system takes the following actions:

**a)** Resets number of blocks (parameter NBLK) to zero.
**b)** Clears option specifications of *all* pulser interval parameters.
**c)** Restores original display & access flags of all parameters to their default values.
**d)** Hides *all* pulser interval parameters.

During the initialization process, the "hidden" flag of *every encountered parameter* (including the ones appearing within expressions) is automatically removed which means that the parameter shall become visible. The rule is that when a parameter is used, it should be displayed, regardless of whether its value is User-accessible or not (unless, of course, its default display location (specified in the file Parameters.def) is null.

## V. The important ENDS parameter

The #SETUP section includes almost always the parameter **ENDS** with the initial settings of the ***receiver phase cycle*** (see the Receiver Phase chapter in the on-line manual).

Example of a #SETUP section:

**D0** :(RD)
**PW** = 90 [p(x,-x,y,-y)]
**ENDS** [p(x,-x,y,-y)]

## VI. The SEQUENCE Section

The sequence section of a pulser sequence script is scanned every time the virtual pulser (i.e., the *image* of the real pulser) is programmed. The order-of-appearance of this section's items is absolutely essential since it defines the sequence of individual pulser steps during the pulse sequence execution.

Each sequence section command may be one of the following:

**1)** *Acronym of a pulser interval parameter* (see *Pulser Interval Parameters* in the on-line manual.
**2)** *Special pulser sequence directive*, preceded by the **#** character.

A *parameter acronym* defines a pulser step whose *pulser channels are defined by the specified parameter's options* (see the *Pulser Parameters Options* chapter in the on-line manual).

The *duration of the step is normally defined by the current value of the parameter*. However, if the parameter is followed by an expression (enclosed in parentheses), the expression is evaluated and the step's duration is set equal to the result ( use this very special override of the displayed parameter value only exceptionally when there is really no other way to achieve your goal).

Within this general framework, some parameters are associated with **special default actions** listed in the following Table

| **Parameter** | **Action** |
|---|---|
| RF pulses (**PW**,**P1**,**P2**,...) | Turn ON the T-channel (transmitter) |
| **D0**, **ACQD**, **STIM** | Turn ON the R-channel (receiver) |
| **D0** | Sets pulser into ARMED state (idle but with active output) |
| **PW** | Resets the time origin (automatic trailing #TIME0 directive) |
| **STIM** | Triggers a sweep (automatic prior #SWEEP directive) |
| **STIM** | Sets XOFF (time-scale offset of the sweep) |

## V. The important D0 step

*Each sequence MUST start with the pre-scan delay D0 which MUST NOT appear elsewhere.*

Remember that once the instrument has been 'updated', **the D0 settings are effective also while the instrument is idle**. For example, should you specify the magnet to be on the polarization field in the D0 period (by means of the m(P) options), it would be really ON and it might BURN! The normal setting of the D0 options is null (everything OFF).

## VI. Pulser sequence #Directives

Pulser sequence directives may appear *only* within the sequence section of a pulser sequence script. They are distinguished from NMR parameters by being **preceded by the # sign** and *interpreted* by the system during pulser programming in order to implement special actions such as loops, pulser-controlled data-acquisition (ADC strobes), etc.

A related special feature are the **pulser programming registers** distinguished by the # sign, followed by a numeric index (#0, #1, ..., #99). These can be assigned numeric values, operated upon and appear as arguments in expressions just like constants and numeric system parameters. This makes it possible to carry out any *User-defined calculations during pulser programming* and even retain memory of the calculated values.

An important set of special directives are the **#label**, **#GOTO** and **#IF** directives which enable the User to implement **any type of looping** (possibly with selected parameter values varying according to User-defined mathematical rules).

Another set of directives, namely **#ADC** and **#SWEEP**, give the user the possibility to control data acquisition strobes using freely either **isolated, pulser-generated ADC strobes** or the **sweep-generator engine** which makes part of the Aqm96 board.

With all these features, complemented by the **parameter options scheme**, the **phase cycling scheme**, and the **X-devices scheme**, it is likely that the pulser system can accommodate essentially any pulse-sequence one can imagine either at present or in the foreseeable future.

 

## VII. Table of currently implemented pulse sequence directives

| Directive | Action |
|---|---|
| **#<index>** = <expression> | Assignment of a pulser programming register |
| **#<label>:** | Target location for the GOTO <label> directive.<br><label> is any User-defined string. |
| **#GOTO** <label> | Unconditional pulser program jump |
| **#IF** <relation> <command> | Conditional execution of a sequence <command><br>(a pulser interval parameter or a directive).<br><relation> is a construct of the form<br><expression><relation operator><expression>,<br>where <relation operator> is one of <,=,>,<=,>=,<>,><. |
| **#TIME0** | Sets the time-axis origin for subsequent data points |
| **#ADC** | Generates an ADC strobe. When at least one #ADC<br>is present, the experiment type (EXPT) is set to to 1<br>(explicit data-points timing array) instead of 0 (regular timing grid). |
| **#SWEEP** | Generates a sweeper strobe. |
| **#SETUP** | Ends SEQUENCE section and starts SETUP section |