

XRD Computing Guide

PINCER Primer

M.C.Miller

SR Facility Computing Group

Daresbury Laboratory

This page intentionally left blank

Table Of Contents

| | |
|--|-----------|
| 1. Introduction | 6 |
| 1.1 Intended Readership | 6 |
| 1.2 The Pincer Document set | 6 |
| 2. The Principles of Pincer Data Acquisition | 7 |
| 2.1 Introduction to Pincer and macros | 7 |
| 2.2 Pincer versions available with features | 7 |
| 2.2.1 Labwindows Pincer32 v8.3 for Windows 95/NT..... | 7 |
| 2.2.2 Labwindows PincerV v7.95 for Windows 3.1/95..... | 8 |
| 2.2.3 PincerW v7.81 for Windows 3.1/95..... | 8 |
| 2.2.4 MS-DOS Pincer v7.81..... | 8 |
| 3. Installing the Pincer program | 9 |
| 3.1 Distribution directory structure and files | 9 |
| 3.2 Quick start | 9 |
| 3.3 Running Pincer test macros | 10 |
| 4. Installing the macro sets | 11 |
| 4.1 PD macros | 11 |
| 4.1.1 Quick start | 11 |
| 4.1.2 Setting up configuration file pdinit.mac..... | 11 |
| 4.1.3 Changing settings in file pdconfig.dat..... | 13 |
| 4.2 CLAM macros | 14 |
| 4.2.1 Quick start | 14 |
| 4.2.2 Running the test macro “clamst.mac” | 14 |
| 4.2.3 CLAM on-line help (“help clam”)..... | 14 |
| 4.3 4CIRCLE macros | 15 |
| 4.3.1 Quick start | 15 |
| 4.3.2 Using the macros with CLAM macros..... | 15 |
| 5. Installing instruments | 16 |
| 5.1 Data acquisition hardware types | 16 |
| 5.1.1 Introduction | 16 |
| 5.2 Adding new instruments to the system | 17 |
| 5.2.1 Adding a new motor to the system..... | 17 |
| 6. Optimising the system | 19 |
| 6.1 Customising startup file startup.mac | 19 |
| 6.2 Customising tidyup file tidyup.mac | 19 |
| 6.3 Multiple macro execution directories | 19 |
| 6.4 SRS data file configuration file location | 19 |
| 7. References | 20 |

This page intentionally left blank

Abstract

This guide is intended primarily to provide introductory information for the operation of Inorganic X-Ray Diffraction (XRD) stations and others which are running the already configured *PINCER*[1] **P**ortable **I**nstrument **C**ontrol interpreter software.

For users outside the Daresbury SRS, the guide should provide enough information to install, configure and use the *PINCER* program and macros in a basic mode.

1. Introduction

1.1 Intended Readership

This guide seeks to provide basic information required for installation and operation of the Pincer program. It does not provide full information of user functions or the various configurations implemented at the SRS and collaborating Universities. These aspects are covered by other guides (see below) which are available in both PDF and HTML format. This is indicated in the URL specified in the footnote of this document and similarly in the other documents.

1.2 The Pincer Document set

- *PINCER Primer*, M.C.Miller, August 1998.
- *PINCER User Guide*, M.C.Miller, C.Marshall and H.Millington, August 1998.
- *PD Macro User Guide*, M.C.Miller, C.C.Tang and E.J.Maclean, August 1998.
- *CLAM and 4circle Macro User Guide*, M.C.Miller and S.P.Collins, August 1998.
- *PINCER System Guide*, M.C.Miller, August 1998.
- *PINCER Configuration Guide*, M.C.Miller, August 1998.

See <http://www.dl.ac.uk/SRS/XRD/pincer.dir> for html and pdf on-line versions

2. The Principles of PINCER Data Acquisition

2.1 Introduction to PINCER and macros

The PINCER program provides a language for data acquisition. Functions are available for instrument control, screen and file i/o, display graphics, maths functions, internal variable manipulation, flow control, and system services. These building block functions can be entered to the command interpreter as single command lines or as blocks of commands executed in loops or conditionally upon the state of the internal variables which have been declared. Input is either from the keyboard or from *macro* command files which combine the same PINCER commands in a file named *<macro name>.mac* and typing just *macro name* to the command prompt will execute all statements within the file. Recently, the ability to input commands from a TCP/IP network socket has been added allowing operation from a de-coupled GUI client program.

Standard macro sets are freely available for use with PINCER which can be readily configured to drive a wide range of instruments. The *PD* macros support menu based positioning and scanning operations and is widely used at Daresbury. The *CLAM* macros represent the next generation of flexibility where users can rapidly write their own “pick and mix” instrument drivers called *Virtual Motor* or *VM* macros. As part of this suite the *4CIRCLE* macros can be added allowing 4 circle diffractometer angle calculations to be included in VM’s along with reflection searching, centring and manipulation.

For more details of the PINCER command syntax, see the *PINCER User Guide* and on-line help from the program.

2.2 PINCER versions available with features

2.2.1 Labwindows PINCER32 v8.3 for Windows 95/NT

This is the flagship version of PINCER most tailored to the Windows environment. It is a full 32-bit program and supersedes the other program versions. It is built with the 32-bit National Instruments Labwindows C compiler. It includes in the graphical user interface a command history window allowing previous commands to be recalled and edited if required. Also the graphics window has its own pull-down menus allowing such operations as printing, toggling of a screen coordinate request cursor and auto-rescaling on the x and y axes. All instrument types are supported and further information can be obtained from the *PINCER User Guide*.

Contact the author to discuss the availability of this program.

2.2.2 Labwindows PINCERV v7.95 for Windows 3.1/95

This is the version of PINCER built for the 16-bit Windows environment. It is built with the National Instruments Labwindows C 16-bit compiler. It includes most features of the 32-bit program but not the TCP/IP server mode.

2.2.3 PINCERW v7.81 for Windows 3.1/95

This program runs as a true Windows application but was built using the Microsoft C/C++ ANSI C QuickWin libraries and so lacks some of the functionality of the other PINCER versions. In particular just the PC COM port RS232 i/o is not available for any devices. All GPIB i/o is available however which includes any GPIB-RS232 converter units which may be purchased freely.

2.2.4 MS-DOS PINCER v7.81

This is the original program environment which requires about 400Kb of conventional DOS memory to run. It should also run as a DOS box under windows if the "exclusive" mode is selected in the properties dialogue box and full-screen mode is selected for the display. As Windows attempts to multitask, problems still may result when instrument i/o is interrupted by a Windows event. The graphics display operations share the same physical screen and so macros such as *CLAM* cannot display all the data to the text screen which it can when running under Windows.

3. Installing the PINCER program

3.1 Distribution directory structure and files

The PINCER and macro set distribution directories either on **xrdsv1** in **~mem/macros.dir** or from a supplied zip file are structured from the top level directory **distrib.dir** as follows:

| Directory | Major Contents |
|-------------|--|
| pincer.dir | Pincer executable program files *.exe hardware configuration files *.dat startup.mac default startup macro tidyup.mac default tidyup macro on error testuic.mac test macro for keyboard input commands testcmd.mac test macro for other commands testmath.mac test macro for maths functions testgra.mac test macro for graphics operations |
| data.dir | *.fon font file(s) for graphics statn.dat station ID file for PINCER <i>srs</i> command runnum.dat last run number file for PINCER <i>srs</i> command *.dat configuration files for PD and 4CIRCLE operation |
| clam.dir | control and sample VM's for CLAM operation clamst.mac CLAM test macro *.dat optional CLAM configuration files |
| pd.dir | PD macro full set |
| 4circle.dir | 4CIRCLE macros for use with CLAM hklmot VM and including aligning and reflection handling |

3.2 Quick start

The easiest way to use the program and macros is to copy all specified files to the default directory and work from there. There are quite a large number of files particularly in the PD macro set and so these can be stored and accessed elsewhere by the use of DOS environment variables **RAMPATH** and **CLISYS**. For details see the *PINCER System Guide* in the section under *PINCER data acquisition software setup*.

For a quick start carry out the following :

1. copy all files from pincer.dir to the current working directory (cwd)
2. copy all files from data.dir to the cwd

You should now be ready to execute the test macros or install the macro sets.

3.3 Running PINCER test macros

The test macros do not test all available commands completely but provide a check that the installation is a successful one. Also by studying the screen and log file output they can be quite useful as examples of using the PINCER command language. Carry out the following procedures to run the test macros :

1. Under DOS, run the dummy PINCER program by typing *pincerd*. This has dummy hardware i/o and so will not try to access any real instrumentation at any point. Under Windows the corresponding program *pincerwd.exe* will need to be run from the program manager or equivalent. Ideally an icon or shortcut should be permanently set up for it (see Windows documentation for information).
2. A prompt for the graphics name is given at startup which should be set to *vga* on DOS systems and *vgw* on Windows systems for correct graphics operation. If the wrong name is given, graphics calls are ignored by PINCER.
3. After some messages a Pincer “>” command prompt should be detected and to this commands and macro names should be supplied.
4. Type the test macro name *testgra*. The graphics mode should be entered to display a title “Poly3 Psychodelic Demo” and a series of polynomial plots should be drawn in a variety of colours. The font file is required for the text, number and x-y marker symbol and if this is absent only an empty box will appear on the display. After the display hit <return> (and ensure that the PINCER standard input / output windows is selected in the case of the windows version).
5. Type the remaining test macro names and follow instructions, running them in the order *testcmd*, *testmath* and *testuic*. The first 2 test macros do not contain any required input from the user and so are the easiest to try first.
6. *tstcmd*, *testmath* and *tstuic* each produce a log file of results in a file of extension *.txt* and this can be compared with an older version called *.txo*. To help with this a DOS program called *diff.exe* has been supplied which can be used as in the *testcmd* example below :

```
diff testcmd.txt testcmd.txo
```

This echoes any differences between the two file which should only correspond to functions which return the time or write SRS data files with a different *r<number>.dat* name derived from last number stored in the file *runnum.dat*.

It should be noted that *diff.exe* is a GNU project tool and is only released here for your private use.

4. Installing the macro sets

4.1 PD macros

4.1.1 Quick start

1. copy all the files from the *pd.dir* directory to the cwd.
2. restart PINCER and then type *PD* to enter the PD menus.
3. The scan, drive and mono operations can be carried out with the dummy motors a and b and further details can be found in the *Station 2.3 manual*.

4.1.2 Setting up configuration file *pdinit.mac*

The instrument parameters in *pdinit.mac* by default are correct for the dummy PINCER operation using PD. For real systems settings will need to be modified and the table below explaining the existing values should give enough information for this.

| Line in default <i>pdinit.mac</i> | explanation |
|--|--|
| <code>if (nargin!=0 nargout!=0)</code> | using inbuilt variables <i>nargin</i> and <i>nargout</i> check for any input and output variables in call to <i>pdinit</i> |
| <code>argerror</code> | invalid argument(s) given in <i>pdinit</i> call so call error handling macro <i>argerror.mac</i> |
| <code>end</code> | end of “if” command block |
| <code>pdclear</code> | remove all PD variables by calling macro <i>pdclear</i> |
| <code>global\$ stn_name="Station tst"</code> | declare global string variable <i>stn_name</i> which is used for graphics title |
| <code>global\$ current_motor="a"</code> | declare global string containing default motor name for “drive” menu |
| <code>global\$ current_mono="SI"</code> | declare global string containing name of mono type used in “mono” menu |
| <code>global\$ valid_motors="a b"</code> | declare global string containing list of motors names separated by a space. Used in PD drive and scan. Must be set in mot parameter files also (see below) |
| <code>global\$ motor_units="stp deg"</code> | declare global string containing a unit name list for each motor name in <i>valid_motors</i> string. Should |

| | |
|---|---|
| | match units set in mot_gen.dat motor parameter file. |
| global\$ motspeed="Default Speed" | declare global string containing speed setting status string. |
| global setspeed=0 | declare global numeric variable indicating speed setting disabled (0) or enabled (1) |
| global\$ th2th_motors="a b" | declare global string containing 2 motors treated as theta and twotheta in th2th and Nth2th scans |
| global\$ mono_motor="b" | declare global string containing name of mono motor for mono menu |
| global\$ fmtdef="%#9.4f" | declare global string containing default format string for drive motor display (here a total field of 9 characters with 4 decimal places and left aligned) |
| %global\$ pdfmt1="%#9.2f slits rot1 rot2 trans" | commented out declaration of global string containing 1 st alternative format string and motors which should use it (overrides fmtdef list) |
| %global\$ pdfmt2="%#9.3f chi phi" | commented out declaration of global string containing 2 nd alternative format string and motors which should use it (overrides fmtdef list) |
| global\$ timechan="ch1" | declare global string containing name string of timing read channel for read detectors, drive and scans. Must be set in ctr parameter files |
| global\$ monchan="ch2" | declare global string containing name string of 1 st detector read channel for read detectors, drive and scans. Must be set in ctr parameter files |
| global\$ sparechan="ch3" | declare global string containing name string of 2 nd detector read channel for read detectors, drive and scans. Must be set in ctr parameter files |
| global\$ detchan="ch4" | declare global string containing name string of 3rd detector read channel for read detectors, drive and scans. Must be set in ctr parameter files |
| global\$ timerchan="timer" | declare global string containing name string of timer channel for setting count interval. Used in read detectors, drive and scans. Must be set in ctr parameter files |
| global\$ temchan1="ox1" | declare global string containing 1 st temperature controller name. Can be enabled in pdconfig.dat |

| | |
|--|--|
| global\$ temchan2="ox2" | declare global string containing 2 nd temperature controller name. Can be enabled in pdconfig.dat |
| global\$ logfile="pd.log" | declare global string containing path to PD log file name |
| global\$ configFile="pdconfig.dat" | declare global string containing path to PD configuration file pdconfig.dat |
| global\$ grdev="vga" | declare global string containing graphics device name (vga for DOS or vgw for Windows) |
| global srsfile=-1 | declare global internal variable for file i/o |
| global ymax_lim=100 | declare global variable containing default ymax graph limit in plots |
| global nosrsheadprompt=0 | declare global internal variable for file i/o |
| global ctr_plot_chan=4 | declare global internal variable for no. of counter-timer channel to plot in scan graphics |
| global\$ srsfilename="" | declare global internal string to hold last SRS datafile name |
| global pd_options[32] | declare global array which will contain PD parameters read from pdconfig.dat file |
| readconf = pd_options | call macro readconf.mac to load pdconfig.dat parameters to pd_options array |
| | |
| init | call macro init.mac to initialise any hardware needed by PD |
| setspd "slow" | call macro setspd.mac to change speed to slow (only active if enabled elsewhere) |
| | |
| return | return to previous macro back to PINCER prompt |
| | |
| % | comment |
| % valid_motors and valid_motors1 will appear in drive menu and | comment |
| % scan menu. valid_motors2 will not but can be set so as to | comment |
| % have a 2nd drive menu. | comment |
| % | comment |
| % stn_name should be set to length 11 characters for neatness | comment |

4.1.3 Changing settings in file pdconfig.dat

This file stores the internal parameter settings that PD requires at run-time. It contains only values and no comments so is modified using 2 other macros as follows from the PINCER prompt (i.e. outside the PD menus) :


```

% gridn <mot> <start> <step1> <n1> <step2> <n2>    number grid scan (single vm)
% gridcn <mot> <step1> <n1> <step2> <n2>          centred grid scan (single vm)
%
% Other scan macros may be available in the future.
%
% Good luck!

```

4.3 4CIRCLE macros

4.3.1 Quick start

1. copy all files from 4circle.dir to the cwd.
2. The macros can be used via the CLAM control macros (see below). For details of the 4 circle macros functionality, see the appendix.

4.3.2 Using the macros with CLAM macros

The CLAM macros should be installed with the 4CIRCLE macros. There is a sample *four.dat* file which contains example experiment and sample parameters so that valid angle calculations can be performed.

The VM *hklmot.mac* is the 4 circle control macro which operates in reciprocal space units *h*, *k*, *l* and can be positioned and scanned like any other. As well as moving to and echoing the reciprocal lattice units (*rlu*) it displays the real diffractometer angles which are also written to the SRS datfile in scans.

The geometry used in the angle calculations is set by a calculation macro name string read from the file *CalcMode.dat* which is present in the *data.dir* directory. The most common mode is bisecting angle which is carried out by the macro *hkl2bis.mac* and so the sample *CalcMode.dat* contains the string *hkl2bis*.

The following example show CLAM pos and scan operations using the *hklmot* VM:

```

>pos hklmot
  H = 0.99997 rlu
  K = 0.99997 rlu
  L = 0.99996 rlu
  tth = 15.1500 deg
  ome = 7.5750 deg
  chi = 89.1390 deg
  phi = -130.2430 deg

>pos hklmot [1 1 0]
  H      K      L      tth      ome      chi      phi
0.99998 0.99998 0.00000 12.3580 6.1790 54.0520 -93.1790

>scan hklmot [1 0 0] [1.1 0 .1] [2 0 1] t 1
>>> Enter title : test
>>> Enter condition 1 :
>>> Enter condition 2 :
>>> Enter condition 3 :
=== SRS file name: r404.dat
dummyvm is here
=== Angle calculation mode: hkl2bis
  H      K      L      tth      ome      chi      phi      time      Ch2      Ch3      Ch4
0.99999 0.00001 0.00000 8.7300 4.3650 34.4670 -152.2220 1.000 0 0 0
1.10002 -0.00001 0.10001 9.6450 4.8220 38.0790 -156.8550 1.000 0 0 0
1.20004 0.00000 0.20001 10.6260 5.3130 40.8990 -161.0080 1.000 0 0 0
1.29997 0.00000 0.30000 11.6560 5.8280 43.1120 -164.7140 1.000 0 0 0
1.39999 0.00001 0.40000 12.7250 6.3630 44.8630 -168.0160 1.000 0 0 0
1.50004 0.00001 0.50001 13.8240 6.9120 46.2620 -170.9570 1.000 0 0 0
1.60001 0.00002 0.60000 14.9460 7.4730 47.3920 -173.5810 1.000 0 0 0
1.69997 0.00000 0.69996 16.0870 8.0440 48.3130 -175.9270 1.000 0 0 0
1.80002 -0.00001 0.80002 17.2450 8.6220 49.0730 -178.0300 1.000 0 0 0
1.89997 0.00000 0.89996 18.4150 9.2080 49.7050 -179.9210 1.000 0 0 0
2.00004 0.00000 1.00001 19.5980 9.7990 50.2360 -178.3730 1.000 0 0 0

```

5. Installing instruments

5.1 Data acquisition hardware types

5.1.1 Introduction

There are five types of hardware device as follows :

- **mot** - motor drives
- **ctr** - counter-timers
- **mca** - multichannel analysers and generic i/o devices
- **tem** - temperature controllers

Each of these has a generic file containing the name of each device and the type of specific hardware that is involved, plus parameters to allow conversion from user-units to hardware-units and back again (see the configuration files for more details). There are a number of possible specific hardware types within each generic device type each specific device has its own configuration file with information relating to its own control parameters only. All possible devices are summarised below together with the configuration file names.

| generic_device | config_file | device no. | specific_device | config_file |
|----------------|-------------|------------|--|-------------|
| mot | mot_gen.dat | 1 | Harwell 6000 stepper | mot_hw.dat |
| | | 2 | NE9100 CAMAC multiplexer | mot_cam.dat |
| | | 3 | McLennan DC servo/stepper | mot_idb.dat |
| | | (3) | McLennan channel select stepper on 7.6 | mot_idc.dat |
| | | 4 | BEDE Minicam steppers | mot_mic.dat |
| ctr | ctr_gen.dat | 1 | Harwell 6000 | ctr_hwc.dat |
| | | 2 | CAMAC QCR, SCI , CPG | ctr_qcr.dat |
| | | 3 | CAMAC time-frame gen., 32-chan scaler | ctr_tfc.dat |
| | | 4 | BEDE Minicam | ctr_mcc.dat |

| | | | | |
|-----|-------------|---|--|-------------|
| mca | mca_gen.dat | 1 | DL100 Hist. Memory, ADC3512, TDC comb. | Mca_pdl.dat |
| | | 2 | AT-bus Canberra S100 card | mca_can.dat |
| | | 3 | test CAMAC device | mca_tca.dat |
| | | 4 | LeCroy 3512ADC/3588 Hist. Memory | mca_lhm.dat |
| | | 5 | Generic string i/o for RS232 ports, Iotec GPIB-RS232 ports or GPIB devices | mca_cio.dat |
| | | 6 | VME/VXI generic i/o via National Instruments MXI bus | mca_mxi.dat |
| tem | tem_gen.dat | 1 | Oxford Instruments ITC4 cryostat | tem_itc.dat |
| | | 2 | Eurotherm 900 EPC | tem_eur.dat |
| pre | pre_gen.dat | 1 | Denisson Hyd. press PC/strain gauges | pre_hyd.dat |

See the specific configuration files for documentation of parameters and a summary of how the devices actually work.

5.2 Adding new instruments to the system

The stages are the same for all hardware types and the example of adding a new camac motor as a *mot* device is illustrated below.

5.2.1 Adding a new motor to the system

Once a new motor drive is connected to the system, three stages are required to allow it to be accessed by PINCER :

1. edit **mot_gen.dat** to include the new *<name>* and other parameters for the motor. This will include the number of the hardware type and the gearing ratio required (step multiplication factor per user unit).
2. edit the specific hardware parameter file e.g. **mot_cam.dat** for a CAMAC multiplexer motor. This should also have the *<name>* together with all other parameters required by that motor type.

3. If the PD macros are being used, add the motor in **pdinit.mac** to the variable *valid_motors* so that the new motor becomes active.

6. Optimising the system

The sections below attempt to state what is possible and the reader is directed to the *PINCER System Guide* for full details of how to carry out the customisation.

6.1 Customising startup file startup.mac

Any actions which are required at program startup time should be placed in this file except for new hardware initialisation commands which should go in *init.mac*

6.2 Customising tidyup file tidyup.mac

If an error occurs or the user enters <CTRL_A> the current command or macro is aborted and the tidyup macro is called. Thus those commands needed to settle the system to a steady state should be added to this macro. If a captive mode of 1 or above is set (as by the PD macros) then only <CTRL_A> will cause *tidyup* to be called and so the macros currently executing during an error are responsible for taking further action.

6.3 Multiple macro execution directories

Using DOS environment variables RAMPATH and CLISYS, two extra directories for macros can be set up. The search for a macro is in the order RAMPATH, cwd and CLISYS. Care should be taken with duplicate files that the desired file is in fact being executed and not a copy from a higher priority search directory.

If a CLISYS directory exists, the hardware parameter files can also be stored there to keep them safely out of the cwd where they may be accidentally deleted.

6.4 SRS data file configuration file location

To override the cwd storage of *statn.dat* and *runnum.dat*, another DOS environment variable can be set up called SYPATH.

7. References

1. "Portable Data Acquisition Software based on a powerful Command Interpreter and Object Oriented Hardware Control", M.C.Miller, K.Ackroyd and G.Oszlanyi. Presented at ESONE Real-time Data Conference (RTD94), Dubna, Russia 27th - 31st July 1994 and Daresbury Laboratory Preprint DL/CSE/P29E.
2. "SRS Station 2.3 Manual", C.C.Tang, M.C.Miller, E.J.Maclean. CLRC Technical Report DL-TR-98-001, March 1998.
3. "A Novel X-ray Diffractometer to Study the Texture of Materials", C.C.Tang, M.C.Miller, S.M.Clark, M.A.Player and G.R.G.Craib. Accepted for J. Synchrotron Rad. 1996 (in press).
4. "Angle Calculations for 3- and 4- Circle X-Ray and Neutron Diffractometers", William R. Busing and Henri A. Levy (1967). Acta Cryst. 22, 457