ARGONNE NATIONAL LABORATORY
Intense Pulsed Neutron Source

The Data Acquisition Team (December 2001)

John Hammonds, DAS Manager
Alok Chatterjee, DAS Software
Rodney Porter, DAS Hardware, Ancillary Equipment
Richard Como, DAS Hardware, Ancillary Equipment
Chris Piatak, DAS Hardware, Ancillary Equipment

The research is sponsored by the U.S. Department of Energy under contract no. W-31-109-ENG-38.

IPNS Run File Scripting Manual

Chapter 1

GETTING STARTED

Chapter 2

RUN FILE SETUP USING SCRIPTS

Chapter 1

GETTING STARTED

1.1 Introduction

The manual describes a procedure for the IPNS software package to handle data acquisition functions like setting-up the control parameters, the necessary tables for histogramming, the timefields etc.

Several concepts are fundamental to the use of this software. These include the concepts of 'Runs' and 'Run Files', 'histograms', detector 'Ids', 'grouping' of detectors, and 'time-focusing' of detectors. These concepts will be discussed in more detail in Appendix B.

Central to the IPNS software package is the concept of an individual 'Run'. A 'Run' is defined as a data acquisition operation of the system with a single distinct set of instrument parameters. Each separate Run is assigned a 'Run number'. All the parameters used to set up a Run (sample and instrument parameters, start and stop times for data acquisition, run title, user name, etc.), and the raw histogram data collected during that Run are all gathered into one 'Run File' identified with that Run number.

1.2 Software requirements and setup

The IPNS package and the Command package (part of the ISAW software) are the minimum software requirements for setting up new IPNS Runs using 'scripts'. In addition, a java virtual machine (JDK 1.3, for example) needs to be installed on the system. These 'scripts' are ascii text files that make calls to java operators in the IPNS package and set various input parameters for a new Run. These scripts can be run using the ScriptOperator class present in the Command package by one of the following commands on a command (MS-DOS prompt or xterm) window:

```
java Command.CommandPane
or
java Command.ScriptOperator "name_of_script_file"
```

The 'scripts' can only be run using the above commands if they are present in certain locations on the system. Typically, one would need a simple text file called IsawProps.dat in the userhome directory that contains name-value pairs. One of the entries in this file could be the GROUP_HOME and a defining path for it, for example,

```
GROUP_HOME  /IPNShome/hrmcs/pnssystem
or
GROUP_HOME  /usr/local/ipnsjava
```

Other directories can also contain scripts and can be defined as GROUP1_HOME, GROUP2_HOME, etc. in the IsawProps.dat file. The search order for scripts starts with the userhome directory, and then the group directories.

In summary, the set up would require the following packages installed on the system,

1. The IPNS package
2. The Command package
3. JDK 1.3 from www.java.sun.com
4. IsawProps.dat text file in the userhome directory that has at least a GROUP_HOME attribute and its value (path to a directory where the scripts are kept).

Chapter 2

RUN FILE SETUP USING SCRIPTS

2.1 Scripting overview and levels

'Scripts' can be classified into three levels: 'user', 'scientist', and 'DAS' level scripts. The 'user' scripts would allow the user to modify some specific information in the run file that changes frequently from run to run. A scientist could change the 'scientist' level scripts on a less frequent basis and the 'DAS' level scripts will probably be maintained by the DAS group.

A description of 'scripts' that can be used to set up a new IPNS Run is given here. Various "parameters" in the run file can be set with appropriate values by using the methods in the IPNS.RunfileBuilder.java class (listed in Appendix A).

In any script that sets up a new Run, a RunfileBuilder object needs to be constructed first. This is done by the command,

```
v[0] = RFBWrapper(infileName)
```

in the script where infileName is the desired Run file name, for example, hrcs1942 or test0001. The RFBWrapper returns an IPNS.RunfileBuilder object that is stored as the first element v[0] of an array v. Once the RunfileBuilder object, v[0], has been created we can set various attributes by the command

```
RFBSetter(v[0], "iName", iname )
```

where RFBSetter uses the RunfileBuilder object v[0] to set the iName attribute with the value iname.

Any parameter used in the script can be prompted for through a dialog box if a $ sign is the first letter in the line followed by the parameter name, its data type and a string prompt message. As an example, the username can be prompted by doing

```
$ username   String(Your Name)   Enter Username
```

This would display a dialog box with a text area having the words "Your Name"
next to a prompt "Enter UserName".

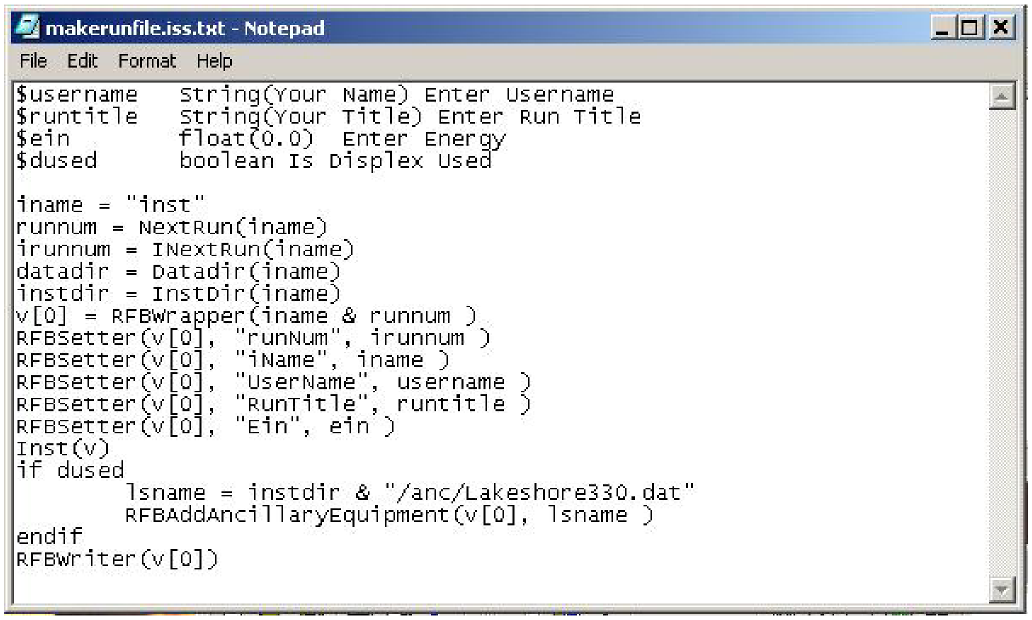
Comment lines can be put in the script by starting the line with a # sign. It is possible to call a script from within another script. This is typically done by passing the original RunfileBuilder object, v[0], to the next script as an argument. An example,

```
Inst(v)
```

is a call to another script that uses the original RunfileBuilder object, v[0], to set other attributes.

2.1.1 'User' level script example

Fig 2-1 shows one 'user' level script that sets up a new run file. It show that it is possible to call scripts from within scripts as shown in the line 'Inst(v)' as discussed earlier. In this "makerunfile.iss" script a number of parameters like the run number (runNum), instrument name (iName), user name(UserName), run title (RunTitle) and the energy in (Ein) for the new run are being set by the RFBSetter method in the RunfileBuilder class. In addition, another script 'Inst(v)' is being invoked to set a different level of parameters and finally the ancillary control parameters are set from a file (lakeshore330.dat) using the RFBAddAncillaryEquipment class and the RFBWriter writes all these inputs into a new Run.

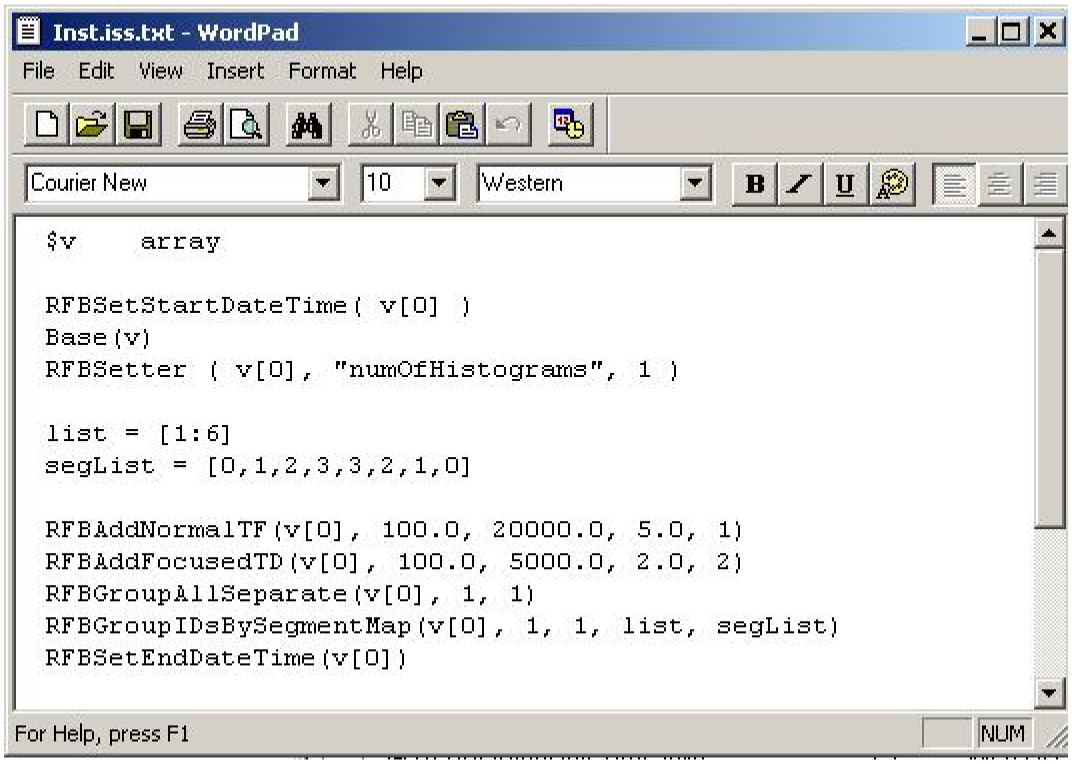


```
$username   String(Your Name) Enter Username
$runtitle   String(Your Title) Enter Run Title
$ein        float(0.0)  Enter Energy
$dused      boolean Is Displex Used

iname = "inst"
runnum = NextRun(iname)
irunnum = INextRun(iname)
datadir = Datadir(iname)
instdir = InstDir(iname)
v[0] = RFBWrapper(iname & runnum )
RFBSetter(v[0], "runNum", irunnum )
RFBSetter(v[0], "iName", iname )
RFBSetter(v[0], "UserName", username )
RFBSetter(v[0], "RunTitle", runtitle )
RFBSetter(v[0], "Ein", ein )
Inst(v)
if dused
        lsname = instdir & "/anc/Lakeshore330.dat"
        RFBAddAncillaryEquipment(v[0], lsname )
endif
RFBWriter(v[0])
```

Figure 2-1. "makerunfile.iss". A 'user' level script.

2.1.2 'Scientist' level script example

In this 'scientist' level script, Fig 2-2, the start date for the run is first set. Next a call to a 'DAS' level script, Base.iss, see Fig 2-3 is made. A new TimeField is added to the Run file using RFBNormalTF and RFBAddFocusedTF.



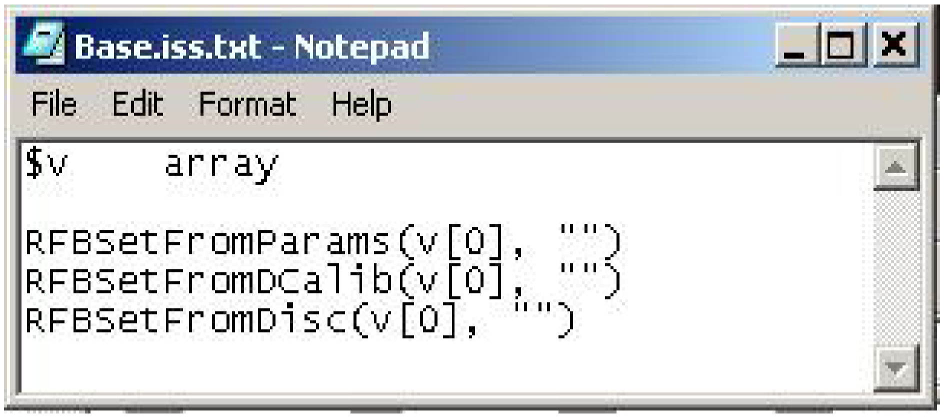
```
$v    array

RFBSetStartDateTime( v[0] )
Base(v)
RFBSetter ( v[0], "numOfHistograms", 1 )

list = [1:6]
segList = [0,1,2,3,3,2,1,0]

RFBAddNormalTF(v[0], 100.0, 20000.0, 5.0, 1)
RFBAddFocusedTD(v[0], 100.0, 5000.0, 2.0, 2)
RFBGroupAllSeparate(v[0], 1, 1)
RFBGroupIDsBySegmentMap(v[0], 1, 1, list, segList)
RFBSetEndDateTime(v[0])
```

Figure 2-2. An Instrument Scientist settable script that call the “DAS level” script, Base(v).

The grouping for the detectors is set using RFBGroupAllSeparate. This makes a separate subgroup for all detector elements. The first integer ‘1’ in the parameter is the time Field to be associated with all detectors. The LPSD’s can have their segments grouped based on a segment map by using RFBGroupIDsBySegmentMap. The end date and time for the run is put in using RFBSetEndDateTime.

2.1.3 DAS level script example

In the ‘DAS’ level script parameters from an instrument parameter file can be set using RFBSetFromParams (RunfileBuilder rfb, String filename). The absolute filepath needs to be present in the filename. It is also possible to provide empty string “” and the default instrument parameter file is located and read in. To have this work the instrument name should be set first by using RFBSetter(rfb, “iName”, “xxxx”), where xxxx is the instrument prefix.



```
$v    array

RFBSetFromParams(v[0], "")
RFBSetFromDCalib(v[0], "")
RFBSetFromDisc(v[0], "")
```

Figure 2-3. A "DAS level" script.

Next parameters from a detector calibration table can be set using RFBSetFromCalib(RunfileBuilder rfb, String filename). The absolute filepath needs to be present in the filename. It is also possible to provide empty string "" and the "default instrument calibration file is located and read in. To have this work the instrument name should be set first by using RFBSetter(rfb, "iName", "xxxx"), where xxxx is the instrument prefix.

Finally, the discriminator values from a file can be set using RFBSetFromDisc(RunfileBuilder rfb, String filename). The absolute filepath needs to be present in the filename. It is also possible to provide empty string "" and the default instrument discriminator file is located and read in. To have this work the instrument name should be set first by using RFBSetter(rfb, "iName", "xxxx"), where xxxx is the instrument prefix.

2.2 Parameter setting order

Generally different attributes can be set in any order in the run file except for a few exceptions. For instance, the call to the RFBWrapper is always the first, followed by RFBSetter (v[0], "iName", inameValue). The iName attribute needs to be set before the Parameter, DetectorCalibration and discriminator files are read in through RFBSetFromParams(v[0],"filename"), RFBSetFromCalib(v[0],"filename"), and RFBSetFromDisc(v[0],"filename") calls. The "numOfHistograms", RFBSetter (v[0], "numOfHistograms", 1) needs to be set before the TimeFields, for instance, RFBAddNormalTF(v[0], 100.0, 2000.0, 5.0, 1), RFBAddFocusedTF(v[0], 100.0, 5000.0, 2.0, 2), and the detector groupings, RFBGroupAllSeparate(v[0],1 ,1), are set. Ancillary equipment settings can be read in from a file by RFBAddAncillaryEquipment(v[0], "filename"). The last call should be to the RFBWritter(v[0]).

2.3 Viewing parameters set in a new Run

It is possible to view the parameter values that have been set using scripts. Once again, the following command on a command (MS-DOS prompt or xterm) window,

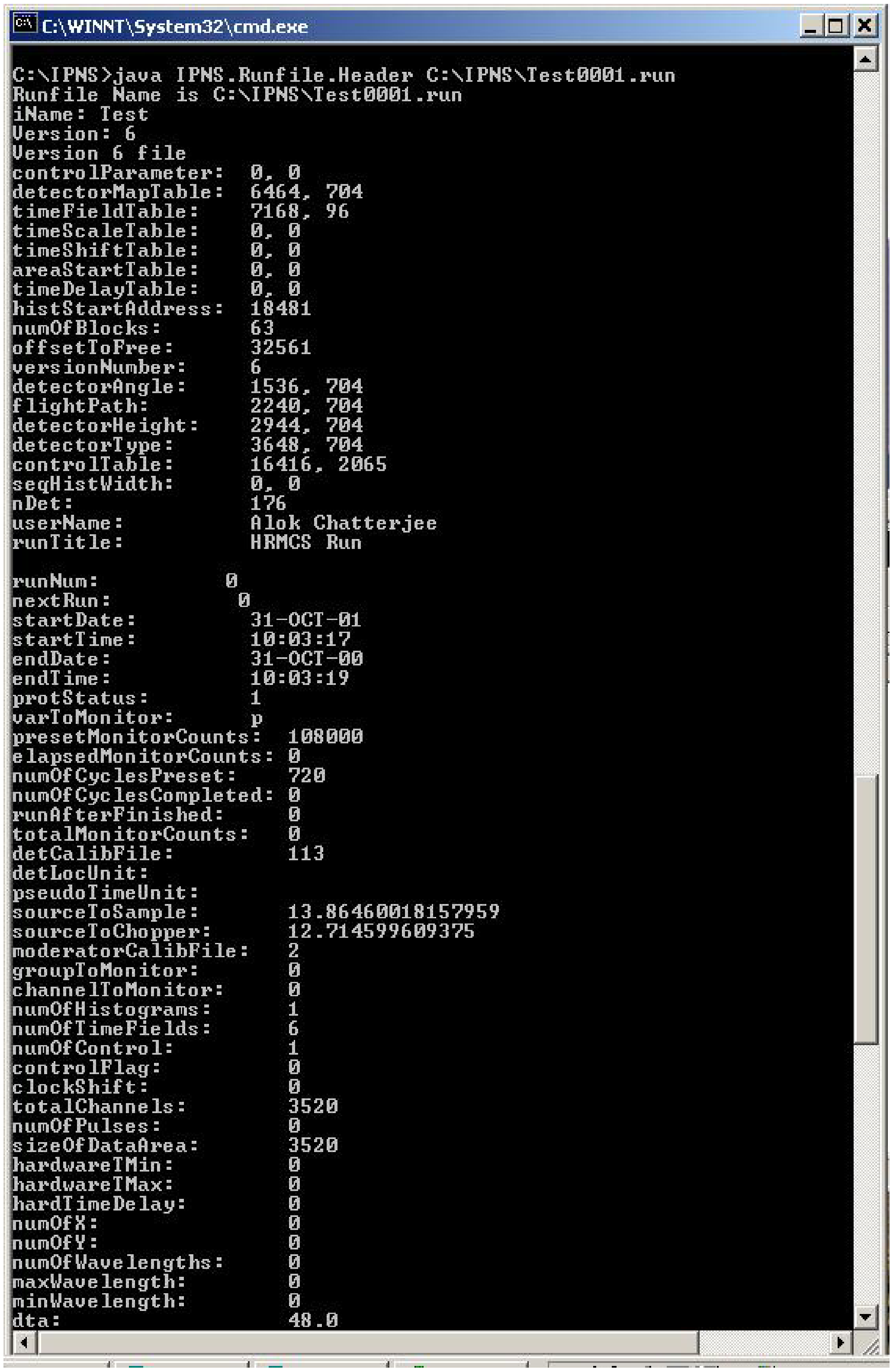
```
java IPNS.Runfile.Header C:\IPNS\Test0001.RUN
```

will display the values stored in the header of a Run file.
To view the time field information use the command

```
java IPNS.Runfile.TimeField C:\IPNS\Test0001.RUN
```

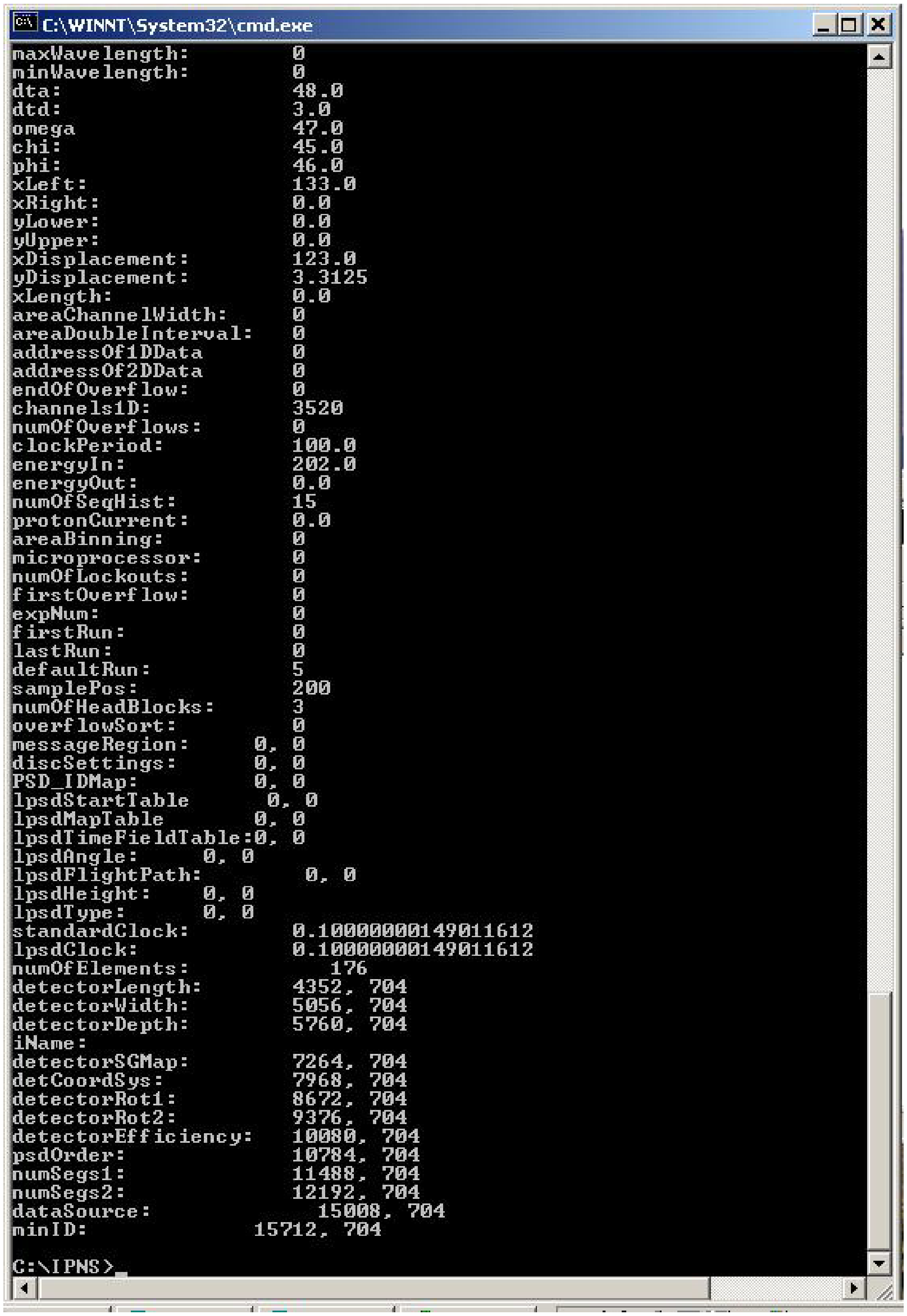
2.3.1 Header file output

Fig 2-4 and Fig 2-5 show the header information from the Test0001.run file that makerunfile script has set up.


C:\WINNT\System32\cmd.exe

```
C:\IPNS>java IPNS.Runfile.Header C:\IPNS\Test0001.run
Runfile Name is C:\IPNS\Test0001.run
iName: Test
Version: 6
Version 6 file
controlParameter:  0, 0
detectorMapTable:  6464, 704
timeFieldTable:    7168, 96
timeScaleTable:    0, 0
timeShiftTable:    0, 0
areaStartTable:    0, 0
timeDelayTable:    0, 0
histStartAddress:  18481
numOfBlocks:       63
offsetToFree:      32561
versionNumber:     6
detectorAngle:     1536, 704
flightPath:        2240, 704
detectorHeight:    2944, 704
detectorType:      3648, 704
controlTable:      16416, 2065
seqHistWidth:      0, 0
nDet:              176
userName:          Alok Chatterjee
runTitle:          HRMCS Run

runNum:          0
nextRun:          0
startDate:         31-OCT-01
startTime:         10:03:17
endDate:           31-OCT-00
endTime:           10:03:19
protStatus:        1
varToMonitor:      p
presetMonitorCounts:  108000
elapsedMonitorCounts: 0
numOfCyclesPreset:    720
numOfCyclesCompleted: 0
runAfterFinished:     0
totalMonitorCounts:   0
detCalibFile:         113
detLocUnit:
pseudoTimeUnit:
sourceToSample:       13.86460018157959
sourceToChopper:      12.714599609375
moderatorCalibFile:   2
groupToMonitor:       0
channelToMonitor:     0
numOfHistograms:      1
numOfTimeFields:      6
numOfControl:         1
controlFlag:          0
clockShift:           0
totalChannels:        3520
numOfPulses:          0
sizeOfDataArea:       3520
hardwareTMin:         0
hardwareTMax:         0
hardTimeDelay:        0
numOfX:               0
numOfY:               0
numOfWavelengths:     0
maxWavelength:        0
minWavelength:        0
dta:                  48.0
```

Figure 2-4. The Header Output after setting up a new Run.



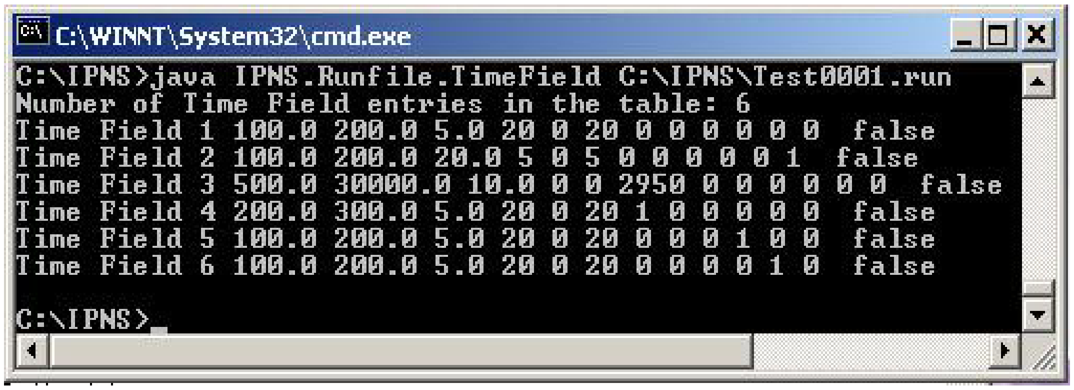
C:\WINNT\System32\cmd.exe

```
maxWavelength:          0
minWavelength:          0
dta:                    48.0
dtd:                    3.0
omega                   47.0
chi:                    45.0
phi:                    46.0
xLeft:                  133.0
xRight:                 0.0
yLower:                 0.0
yUpper:                 0.0
xDisplacement:          123.0
yDisplacement:          3.3125
xLength:                0.0
areaChannelWidth:       0
areaDoubleInterval:     0
addressOf1DData         0
addressOf2DData         0
endOfOverflow:          0
channels1D:             3520
numOfOverflows:         0
clockPeriod:            100.0
energyIn:               202.0
energyOut:              0.0
numOfSeqHist:           15
protonCurrent:          0.0
areaBinning:            0
microprocessor:         0
numOfLockouts:          0
firstOverflow:          0
expNum:                 0
firstRun:               0
lastRun:                0
defaultRun:             5
samplePos:              200
numOfHeadBlocks:        3
overflowSort:           0
messageRegion:       0, 0
discSettings:        0, 0
PSD_IDMap:           0, 0
lpsdStartTable        0, 0
lpsdMapTable         0, 0
lpsdTimeFieldTable:0, 0
lpsdAngle:       0, 0
lpsdFlightPath:         0, 0
lpsdHeight:      0, 0
lpsdType:        0, 0
standardClock:          0.10000000149011612
lpsdClock:              0.10000000149011612
numOfElements:             176
detectorLength:         4352, 704
detectorWidth:          5056, 704
detectorDepth:          5760, 704
iName:
detectorSGMap:          7264, 704
detCoordSys:            7968, 704
detectorRot1:           8672, 704
detectorRot2:           9376, 704
detectorEfficiency:     10080, 704
psdOrder:               10784, 704
numSegs1:               11488, 704
numSegs2:               12192, 704
dataSource:               15008, 704
minID:               15712, 704

C:\IPNS>_
```

Figure 2-5. More Header Output.

2.3.2 Time field output

Fig 2-6 shows the Time field information that also was set up from the script.



```
C:\IPNS>java IPNS.Runfile.TimeField C:\IPNS\Test0001.run
Number of Time Field entries in the table: 6
Time Field 1 100.0 200.0 5.0 20 0 20 0 0 0 0 0 0  false
Time Field 2 100.0 200.0 20.0 5 0 5 0 0 0 0 0 1  false
Time Field 3 500.0 30000.0 10.0 0 0 2950 0 0 0 0 0 0  false
Time Field 4 200.0 300.0 5.0 20 0 20 1 0 0 0 0 0  false
Time Field 5 100.0 200.0 5.0 20 0 20 0 0 0 1 0 0  false
Time Field 6 100.0 200.0 5.0 20 0 20 0 0 0 0 1 0  false

C:\IPNS>_
```

Figure 2-6. Time field output after a new run has been set up.

APPENDIX A

Package description
The IPNS package contains an Operators directory that has all the java classes that one would need to write scripts that set up new runs. A brief description of the purpose and function of each of the these operators is described below:

RFBWrapper (String infileName)
Creates a wrapper object around the IPNS.Runfile.RunfileBuilder class for use in scripts. It returns a RunfileBuilder object that now has access to all public methods in the RunfileBuilder class. This class has to be called first in the script.

RFBWrapper2 (String infileName, String iName, int versionNumber, int instrumentType)
Creates a wrapper object around the IPNS.Runfile.RunfileBuilder class for use in scripts. It returns a RunfileBuilder object that now has access to all public methods in the RunfileBuilder class. It will set the iName, versionNumber and instrumentType as it creates the new runfile. This class has to be called first in the script. To be used instead of RFBWrapper(String infileName).

RFBSetStartDateTime (RunfileBuilder rfb)
Set the start date and time for the run. Makes use of the rfb object provided by the RFBWrapper.

RFBSetter (RunfileBuilder rfb, String attr, Object val)
Set the specified "attribute" attr with the "value" val in the runfile header.
(List of available attributes will be provided later in the documentation).

RFBSetFromParams (RunfileBuilder rfb, String filename)
Set in the runfile the parameters from the instrument parameter file, filename.
The absolute filepath needs to be present in the filename. It is also possible to provide empty string "" and the default instrument parameter file is located and read in. To have this work the instrument name should be set first by using RFBSetter(rfb, "iName", "xxxx"), where xxxx is the instrument prefix.

RFBSetFromDCalib (RunfileBuilder rfb, String filename)
Set in the runfile the parameters from the detector calibration table, filename. The absolute filepath needs to be present in the filename. It is also possible to provide empty string "" and the "default instrument calibration file is located and read in.
To have this work the instrument name should be set first by using RFBSetter(rfb, "iName", "xxxx"), where xxxx is the instrument prefix.

RFBSetFromDisc (RunfileBuilder rfb, String filename)
Set the discriminator values from the file, filename. The absolute filepath needs to be present in the filename. It is also possible to provide empty string "" and the default instrument discriminator file is located and read in. To have this work the instrument name should be set first by using RFBSetter(rfb, "iName", "xxxx"), where xxxx is the instrument prefix.

RFBAddNormalTF (RunfileBuilder rfb, Float min, Float max, Float step, Integer TFNum)
Adds a new TimeField to a Runfile. Here min is the minimum time for the field, max is the maximum time for the field, step is the step size for the field and TFNum is the Time Field Number.

RFBAddFocusedTF (RunfileBuilder rfb, Float min, Float max, Float step, Integer TFNum)
Adds a new TimeField to a Runfile. Here min is the minimum time for the field, max is the maximum time for the field, step is the step size for the field and TFNum is the Time Field Number.

RFBAddWavelengthTF (RunfileBuilder rfb, Float min, Float max, Float step, Integer TFNum)
Adds a new TimeField to a Runfile. Here min is the minimum time for the field, max is the maximum time for the field, step is the step size for the field and TFNum is the Time Field Number.

RFBAddPulseHeightlTF (RunfileBuilder rfb, Float min, Float max, Float step, Integer TFNum)
Adds a new TimeField to a Runfile. Here min is the minimum time for the field, max is the maximum time for the field, step is the step size for the field and TFNum is the Time Field Number.

RFBGroupAllSeparate (RunfileBuilder rfb, int tf, int hist)
Set the grouping for the detectors. Makes a separate subgroup for all detector elements. tf is the time Field to be associated with all detectors.

RFBGroupIdsSeparate (RunfileBuilder rfb, int tf, int hist, int[] list)
Set the grouping for the detectors. Makes a separate subgroup for all Ids listed. tf is the time field to be associated with all detectors and int[] is the list of detector Ids.

RFBGroupIdsSeparateByAngle (RunfileBuilder rfb, int tf, int hist, float[] lowerValue, float[] upperValue)
Set the grouping for the detectors by angle. Returns an integer array of IDs in a specified detector angle range. tf is the time Field to associate with all detectors and hist the histogram that the ids will be binned in. lowerValue and upperValue are the lower and upper limits of the angles to be considered.

RFBGroupIDsBySegmentMap (RunfileBuilder rfb, int tf, int hist)
Makes a separate subgroup by segment map. Here tf is the time Field to associate with all detectors and hist the histogram that the ids will be binned in.

RFBAddAncillaryEquipment (RunfileBuilder rfb, String filename)

Set the information about the ancillary control values from the file, filename. The absolute filepath needs to be present in the filename.

RFBModifyHeaderElement (RunfileBuilder rfb, String attr, Object val)
Modify the specified "attribute" attr with the "value" val in the runfile.

RFBSetEndDateTime (RunfileBuilder rfb)
Set the end date and time for the run. This should the last call before the runfile is written out.

RFBWriter (RunfileBuilder rfb)
Will write out the runfile.

Instdir (String iname)
Returns string with InstrumentDirectory for inst string iname.

Datadir (String iname)
Returns string with DataDirectory for inst string iname.

NextRun (String iname)
Returns formatted string with run number for next run to be created for inst.

InextRun (String iname)
Returns integer with run number for next run to be created for inst.

IncNextRun (String iname)
Increments the last run in parameter file

RawDASLoad (String infileName)
This operator reads data from an ASCII text file and stores the data stored in a DataSet. The data format that is read is basically just x, y pairs in columns with some preliminary information lines.

APPENDIX B

TimeField Table output

tMin, tMax, tStep, NumOfChannels(), timeFocusBit, emissionDelayBit, constantDelayBit, energyBinBit, wavelengthBinBit, pulseHeightBit, Boolean used