Where is it?

How can I get to it?

Where did it come from?

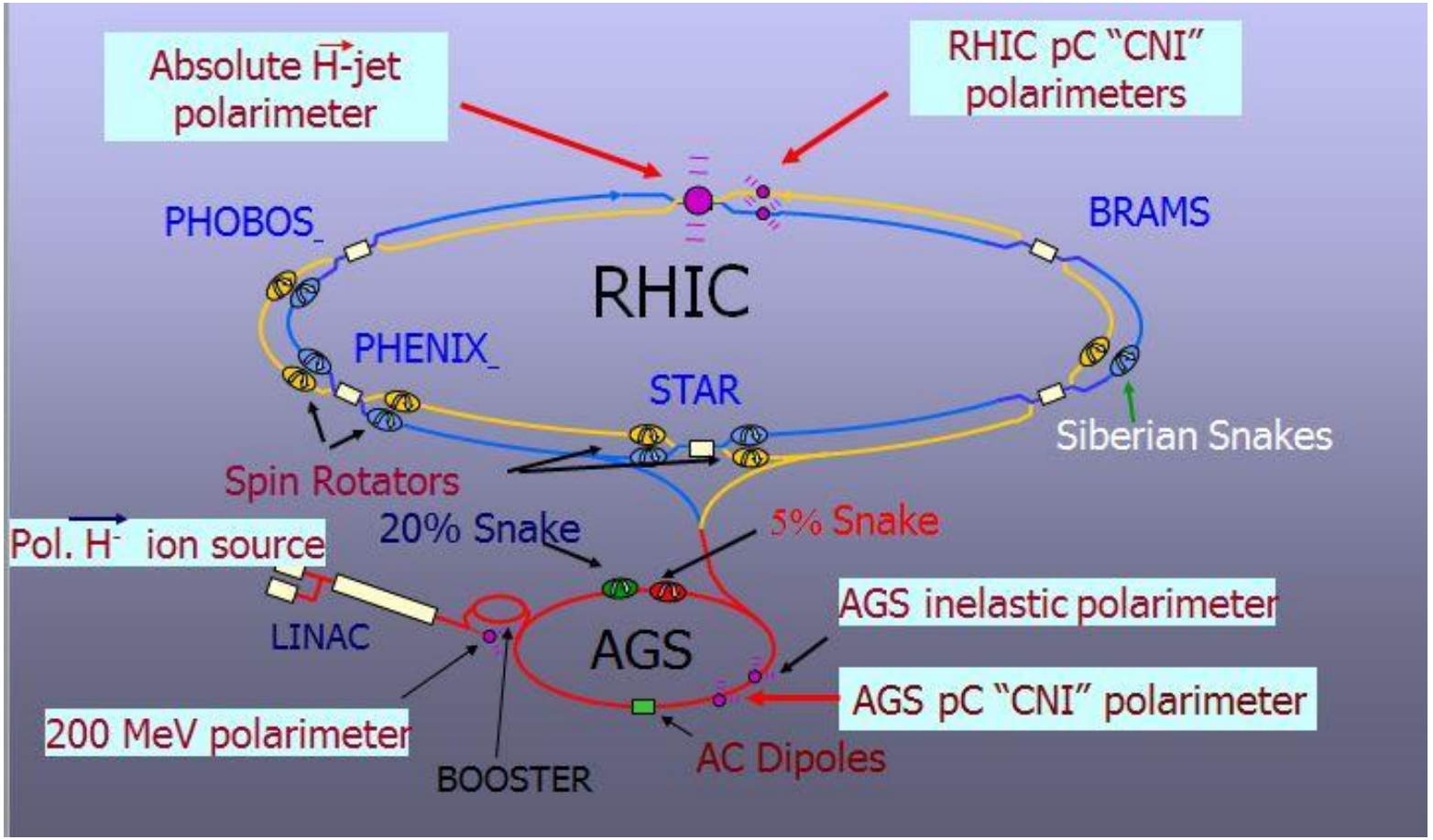
Spin Information

How did we get it?

How should I use it?

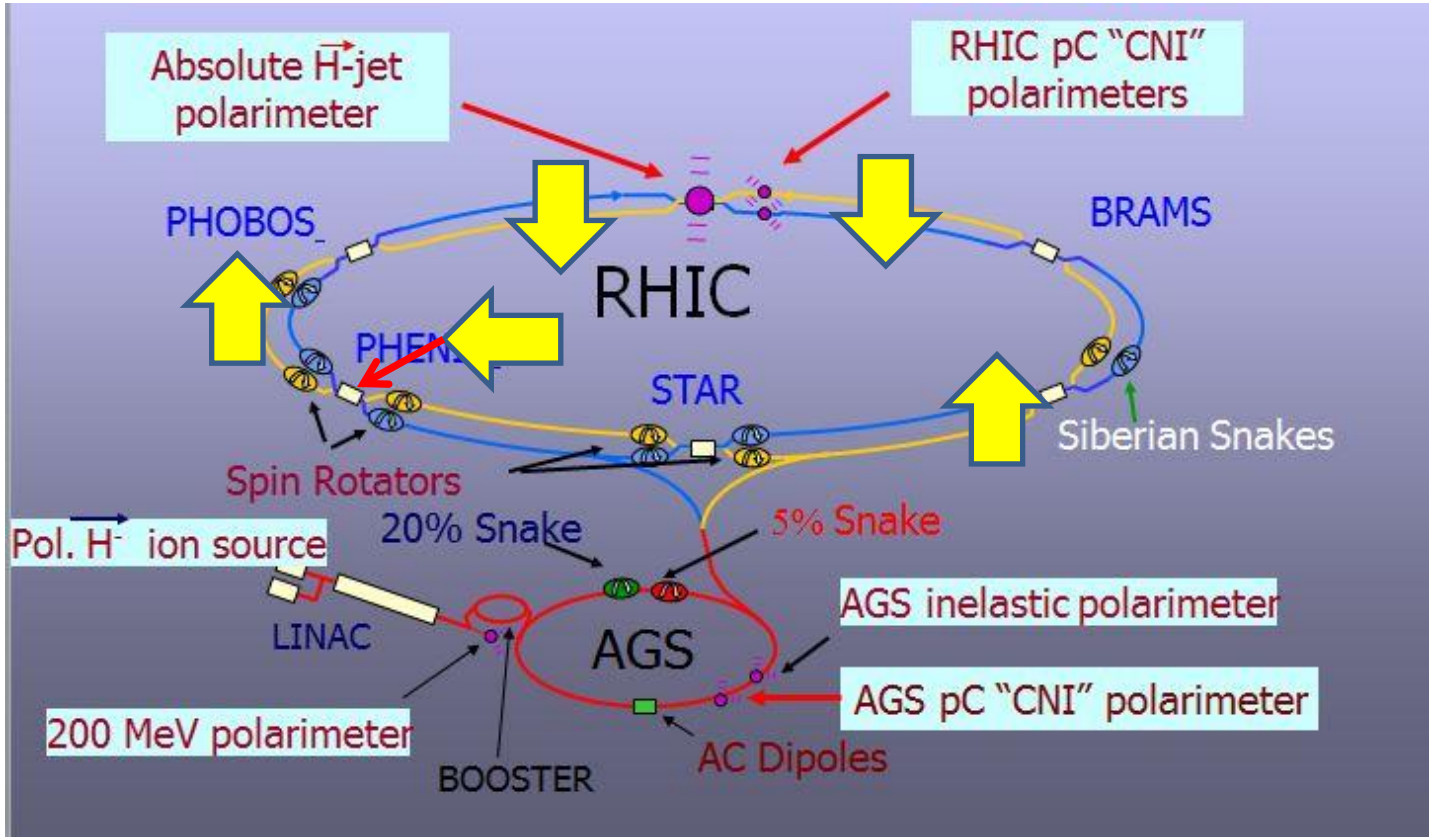
Ciprian Gal
Stony Brook University

RHIC polarized proton journey



Anatoli Zelinsky_PolarizedSource Talk

RHIC polarized proton journey



May Bai may.m.bai@gmail.com via stonybrook.edu 8/3/12

to Ciprian

Hi Ciprian

Vahid, Vadim and I went through the design documents and some trackings. Based on what we have, we believe that the RHIC rotators are configured to give proton with spin up a positive helicity at STAR and negative helicity at PHENIX. This applies to both beams.

Hope this helps.

Best,

Mei

Getting the information

- There seem to be two ways to get the polarization information from collected data:
  - GL1 packet
  - CDEV packet

Getting the information: GL1

- Polarization wise the most important is the GDISABLE ‘word’
- This gives us the V124 pattern

```
int lpAccessor::GetGL1(Packet *p){

  // converting crossing id (360 -> 120)
  iCrossing = ( p->iValue(0,"CROSSCTR") + shiftCrossing ) %nCrossing;
  gdisable = (unsigned long)(p->iValue(0,"GDISABLE"));
  scaled_trig = p->iValue(0,"SCALEDTRIG");
  trig = p->iValue(0,"TRIG");
  rbits0 = p->iValue(0,"RBITS0");
  rbits1 = p->iValue(0,"RBITS1");

  rawtrigword    = p->iValue(0,"RAWTRIG");
  livetrigword   = p->iValue(0,"LIVETRIG");

  // Calclulate Spin Bit
  int bspin1=0,yspin1=0;

  if((gdisable&0x00400000)>0){yspin1= 1;}
  if((gdisable&0x00800000)>0){bspin1= 1;}
  if((gdisable&0x01000000)>0){yspin1=-1;}
  if((gdisable&0x02000000)>0){bspin1=-1;}

  if(bspin1==1 && yspin1==1){spinstate=0;}
  else if(bspin1==1 && yspin1==-1){spinstate=1;}
  else if(bspin1==-1 && yspin1==1){spinstate=2;}
  else if(bspin1==-1 && yspin1==-1){spinstate=3;}
  else{spinstate=4;}
```

Getting the information: CDEV

```
FillNumB = (int) p->dValue(0,"ring_fillNumberB");
FillNumY = (int) p->dValue(0,"ring_fillNumberY");
for(int i=0;i<120;i++)
  {
    Bint[i] = static_cast<int>( p->dValue(i*3,"bkt_intendedFillPatternB") );
    Bpol[i] = static_cast<int>( p->dValue(i*3,"bkt_polarizationFillPatternB") );
    Yint[i] = static_cast<int>( p->dValue(i*3,"bkt_intendedFillPatternY") );
    Ypol[i] = static_cast<int>( p->dValue(i*3,"bkt_polarizationFillPatternY") );
  }

for(int i=0;i<120;i++){
  if(Bint[i]==0) PolPatB[i]=10;
  else if(Bpol[i] == 0 ) PolPatB[i]=0;
  else if(Bpol[i] == 1 ) PolPatB[i]=1;
  else if(Bpol[i] == -1 ) PolPatB[i]=-1;
  else PolPatB[i]=10;
```

- The CDEV packet is filled by CAD and is transmitted to the experiments
- There are 3 pieces of information you can get from this
  - bkt_intendedFillPattern[B/Y] -> what the script that CAD runs tells the source to do
  - bkt_polarizationFillPattern[B/Y] -> what the polarimeters measure as going through the machine
  - Bkt_measuredFillPattern[B/Y] -> what the measured pattern coming out of the source is (V124 – maybe!?)

Xingshift Calibration

- Once we have a spinPattern we can use our triggers to determine where the first bunches are in our data (xingshift) -> In this example 5

-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,10,10,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,-1,-1,1,-1,1,1,-1,1,10,10,10,10,10,10,10,10,10}

8252470,8742050,8848541,8920474,9724611,9445458,9522433,9391548,9517870,8940619,9102941,9929439,9629616,9593346,9060151,9629568,9591048,9461473,9837755,9736940,9889509,9862106,9749725,9762996,9938586,10121104,10077938,10198562,9953582,9147982,9994857,9932393,10442060,369,334,9636976,10436595,10446074,10404088,10390405,10265010,10529087,10300964,9193082,10234419,9389094,10376741,10407231,10300147,10388639,10223559,10161735,10368019,10173274,10151636,9936853,10272692,10362197,10152957,10071337,9947824,10230279,374,9275069,9585500,9518164,10488756,9880329,9450473,10087056,9411618,9984216,9837910,1058,994,9734037,10421086,10643117,10310601,10346704,10201951,10249323,10420011,10200369,10260356,10164413,8870606,9992009,9967916,9912876,10011900,10490911,9639280,9535636,9605834,9929126,10128090,10099314,9759096,9718483,10278583,10064880,10087395,10420507,10112991,9977911,54,34,29,89,32,15,21,16,9,6538744,0,7744214,8059408,8230518

- The data in your DST/CNT/MWG has this crossingshift (when you ask for crossing number from your trigger node it will tell you (spinpatternCrossing – 5) %120)

Xingshift Calibration

- Once we have a spinPattern we can use our triggers to determine where the first bunches are in our data (xingshift)
  - It is first attempted with the V124 pattern
  - Then with the intended spin pattern
- The xingshift calibration is performed when we change bufferboxes during the Run and dumps the data in the **spin_oncal** DB
- Code at (have some popcorn handy when going through this):
  - https://www.phenix.bnl.gov/viewvc/viewvc.cgi/phenix/online/calibration/onlcal/subsystems/xingshift
  - https://www.phenix.bnl.gov/viewvc/viewvc.cgi/phenix/offline/packages/xingshift/

Spin DB

- Merged from the **spin_oncal** and **spin_daq** DBs
- Even though both have the same fields they are only partially filled

```
spin=> \d spin_oncal
            Table "public.spin_oncal"
      Column        |   Type    |     Modifiers
--------------------+-----------+--------------------
 runnumber          | integer   | not null
 fillnumber         | integer   |
 badrunqa           | integer   |
 crossingshift      | integer   |
 polarblue          | real[]    |
 polarblueerror     | real[]    |
 polaryellow        | real[]    |
 polaryellowerror   | real[]    |
 spinpatternblue    | integer[] |
 spinpatternyellow  | integer[] |
 bbcvertexcut       | bigint[]  |
 bbcwithoutcut      | bigint[]  |
 zdcnarrow          | bigint[]  |
 zdcwide            | bigint[]  |
 badbunchqa         | integer[] |
 transversxblue     | real      |
 transversxblueerr  | real      |
 transversyblue     | real      |
 transversyblueerr  | real      |
 transversxyellow   | real      |
 transversxyellowerr | real     |
 transversyyellow   | real      |
 transversyyellowerr | real     |
 qa_level           | integer   | not null default 0
Indexes:
    "spin_oncal_pkey" PRIMARY KEY, btree (runnumber, qa_level)
```

Filling the Spin DB

- spin_oncal
  - runnumber
  - fillnumber
  - crossingshift
  - spinpattern
  - badbunchqa
  - badrunqa (generally empty)
  - Polarization (these are the online values)
- spin_daq
  - runnumber
  - fillnumber
  - bbcvertexcut
  - bbcwithoutcut
  - zdcnarrow
  - zdcwide

Filling the Spin DB

- There has been a lot of work done by Paul Kline in the last 3 runs updating the DB and he has developed a very nice package to interact with our spinDB (https://www.phenix.bnl.gov/viewvc/viewvc.cgi/phenix/offline/analysis/SpinDBUpdater/)
- This was mainly developed for updating the spinDB and it's an added layer of abstraction over the uspin library which most of us use
- The nice thing is that it can be used to obtain the values you need from the spinDB (as it has all the getter methods you need)

Filling the Spin DB

Log of changes made to the DB

Changes made to DB

Date	Changes Made By	Runs Affected	Version (qa_level)	Notes
Jan, 2008	Nathan Means	190454 - 197795		Put in final scalar values
March 26, 2008	Todd Kempel	all		Updated uspin software to use qa_level indexing (the spin DB's new versioning system)
March 26, 2008	Ruizhe Yang	190454 - 197795	3	Updated the polarizations with final values from RHIC CNI Polarimetry group[1]. Note that the final polarizations include both statistical and systematic errors, they are stored speparately in the database.
Aug 1, 2008	Ruizhe Yang	256450 - 259576	1	Added run8 online values of polarization, scaler counts, and spin patterns.
Nov 26, 2012	Paul Kline	357665 - 363228	2	Run 12, 200 GeV CoM. As for qa_level 1 (see above), except a fix for an incorrectly applied crossing shift to the GL1p scaler values.
Nov 26, 2012	Paul Kline	364574 - 368798	2	Run 12, 510 GeV CoM. As for qa_level 1 (see above), except a fix for an incorrectly applied crossing shift to the GL1p scaler values.
Mar 11, 2013	Paul Kline	358985, 358986, 358988, 358991, 360474, 360475, 367538, 367598	3	Various fixes. See http://www.phenix.bnl.gov/WWW/p/forms/info/show_note.php?editkey=an1096

Categories: Databases | Run 12

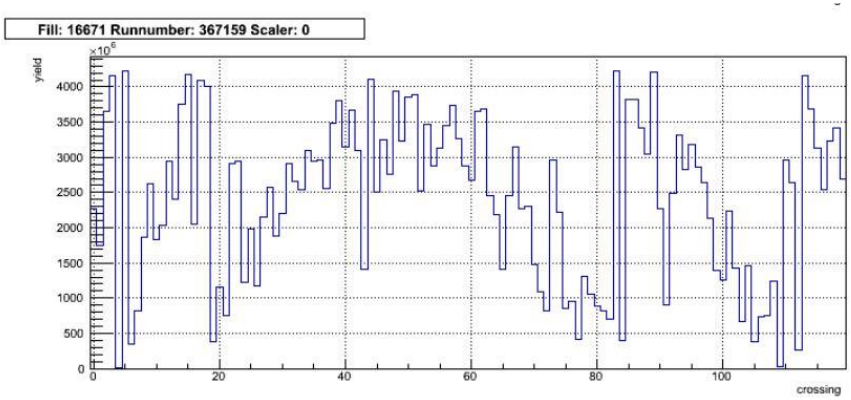
- The qa_level is increased every time the DB is modified
- **All** of the changes to the spin DB should be logged in the offline wiki (https://www.phenix.bnl.gov/WWW/offline/wikioff/index.php/Log_of_changes_made_to_the_DB)

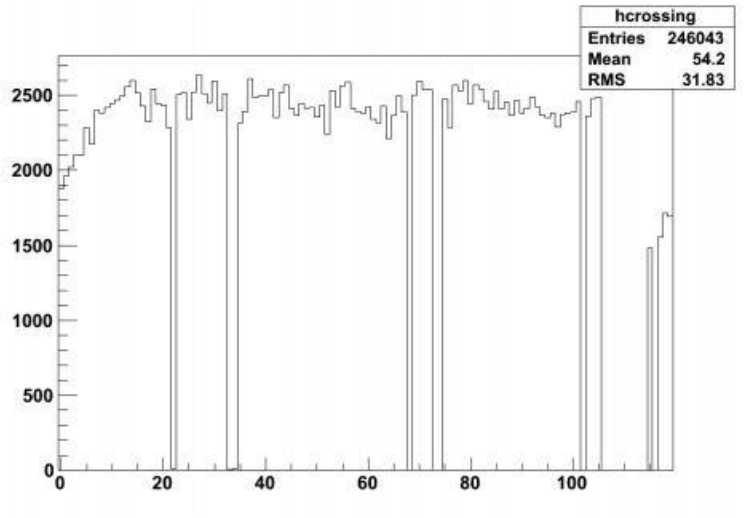
Spin DB QA

- All data is in the DB
- Check that the polarization values in the DB are the same as in the RHIC polarimetry group release
- Scaler values consistent with the GL1p values (and crossingshift)
- Consistency of spin pattern, crossing shift throughout a fill

- https://www.phenix.bnl.gov/phenix/WWW/p/info/an/1096/1096.pdf

Spin DB QA





(d) Run 358992, Shift: 5

- For example:
  - On the left you see random scaler count
  - On the right clearly the xingshift calibration failed
- All the code for the checks we did for run12 are in cvs
- https://www.phenix.bnl.gov/viewvc/viewvc.cgi/phenix/offline/analysis/koster/chkSpinDB/

Getting what you need from the DB

- We have a package called “uspin” (https://www.phenix.bnl.gov/viewvc/viewvc.cgi/phenix/offline/packages/uspin)
- There are some examples on how to use it directly at:
  - https://www.phenix.bnl.gov/WWW/offline/wikioff/index.php/Get_Data_From_Database_(using_root)
  - Example class (the updater):
    - https://www.phenix.bnl.gov/viewvc/viewvc.cgi/phenix/offline/analysis/SpinDBUpdater/

Careful how you use libuspin

- libuspin has normal getter methods but also has PHENIX getter methods
  - The difference is that the ones that are appended with PHENIX at the end take into account the xingshift
  - It is really easy to get confused in how many times the xingshift is applied so I would strongly suggest using the simple getter methods that give you exactly what is in the DB

Small example

```
int tst(int run=340515,int qa=1)
{
  gSystem->Load("libSpinDBUpdater.so");
  SpinDBUpdater *sp = new SpinDBUpdater();

  sp->SetVerbosity(1);
  int exists=sp->GetSpinFields(run,qa);
  if(exists!=-1)
    {
      cout<<"Runnumber:"<<sp->GetRunNumber()<<endl;
      cout<<"Xing shift:"<<sp->GetCrossingShift()<<endl;
      cout<<"Fillnumber:"<<sp->GetFillNumber()<<endl;
      cout<<"BadRunQA:"<<sp->GetBadRunQA()<<endl;
      cout<<"Spin pattern 2(B Y):"<<sp->GetSpinPatternBlue(2)<<" "<<sp->GetSpinPatternYellow(2)<<endl;
      cout<<"Pol blue:"<<sp->GetPolarizationBlue()<<" \pm "<<sp->GetPolarizationBlueStat()<<" \pm "<<sp->GetPolarizationBlueSyst()<<endl;
    }
  else
    {
      cout<<"Cannot find run/qa_level:"<<run<<" "<<qa<<endl;
    }
  return 0;
}
```

- The rest of the getter methods are listed and commented in Paul's code: (https://www.phenix.bnl.gov/viewvc/viewvc.cgi/phenix/offline/analysis/SpinDBUpdater/SpinDBUpdater.h?revision=1.2&view=markup)

```
[ciprian@rcas2064 SpinDBUpdater]$ root -l tst.C
********************************
*  Welcome to ROOT v5.30/03    *
********************************

root [0]
Processing tst.C...
Connected to spin DB.
Initializing SpinDBUpdater
 Table_name is changed from spin to spin
 Table_name is changed from spin to spin_daq
 Table_name is changed from spin to spin_oncal
Runnumber:340515
Xing shift:5
Fillnumber:15472
BadRunQA:0
Spin pattern 2(B Y):-1 -1
Pol blue:0.4244 pm 0.0402 pm 0
(int)0
root [1]
```