

# Peter Willendrup and Jakob Garde DTU Physics

# mcstas-2.x vs. mcstas-3.0, status and elements of the GPU port



### McStas 2.x -> McStas 3.0 main differences

- Rewritten / streamlined simplified code-generator with
  - Much less generated code
  - improved compile time and compiler optimizations, esp. for large instrs
  - changes to engine API for component developers
  - definition parameters no longer supported in components (strings, arrays etc. as replacement)
  - much easier to hack/experiment directly on the generated code
  - new USERVARS feature which enriches the particle struct to enable "per particle" flags
  - Much less invasive use of #define
  - Component sections -> functions rather than #define / #undef
  - Much less global variables, instrument, component and neutron reworked to be structures
- Use of #pragma acc ... in lots of places (put in place by cogen where possible)
- New random number generator implemented
- Complete change to dynamic monitor-arrays
- Various tricks relating to GPU



#### Main events on timeline of current developments

2017: E. Farhi initial cogen modernisation Fall 2018 onwards: J. Garde further cogen modernisation and restructuring

October 2019 onwards: J. Garde & P. Willendrup: New RNG, test system, multiple functional instruments.

March 2018: Participation at Dresden Hackathon. 1st "null" instrument prototype runs.

אופועס 🖉

October 2019: Participation at Espoo Hackathon. First meaningful data extracted. Work on cogen and realising we need another RNG.

November-December 2019: First good look at benchmarks and overview of what needs doing for first release with limited GPU support.

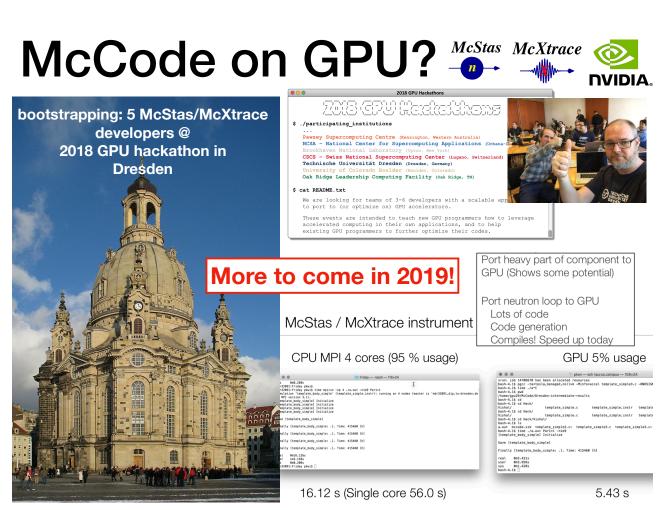
mentor: Christian Hundt 🔁



### McStas heading for the GPU... March 2018

1st prototype, "null"instrument with only one component.

Based on NVIDIA compiler technology, PGCC and OpenACC pragmas





#### McStas heading for the GPU... October 2019

Rewritten codegeneration with automated additions of OpenACC pragmas.

Quite transparent wrt. CPU vs. GPU

First simulations with meaningful output

Speed on DELL with Quadro-card ~ on par with running on CPU with MPI

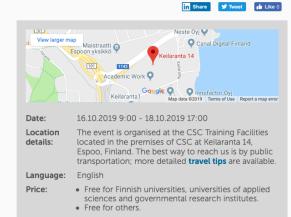
#### **GPU Hackathon**

#### Introduction

CSC is in collaboration with Nvidia and the E-CAM European HPC center of Excellence arranging a 3-day GPU hackathon. The GPU hackathon is a coding event in which teams of developers port their applications or kernels to run on GPUs, or optimize their applications that already run on GPUs. In particular the hackathon focuses on applications that can scale up to multiple GPU nodes.

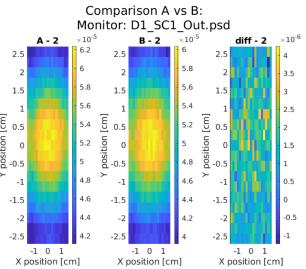
We are looking for teams of 3-4 developers. Collectively the team should know the application intimately. Please keep in mind that we are looking for teams with plans to develop GPU code – not to just run their code on GPUs. During the hackathon each team is supported by one mentor with in-depth GPU programming expertise.

At CSC the new Puhti-Al partition provides 80 nodes with 4 NVidia Volta GPUs each. This system provides in total more than 2 petaflops of performance. This system is available during the course and accepted teams will also have access to the system beforehand to do some initial porting of the applications to Puhti.



The fee covers all materials, lunches as well as





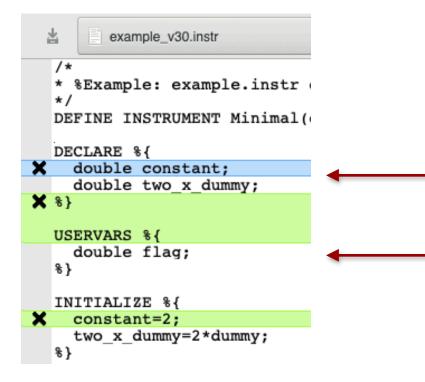




- Illustration, simple instr with
- Instr vars and "flag"
- Arm
- Source
- Slit
- PSD

	a example_v25.instr	<b></b>	example_v30.instr	<u> </u>
N	/*		/*	
N	<pre>* %Example: example.instr dummy=0 Detector: detector_I=345.995 */</pre>		<pre>* %Example: example.instr dummy=0 Detecto */</pre>	r: detector_I=345.995
	DEFINE INSTRUMENT Minimal(dummy=0)		DEFINE INSTRUMENT Minimal(dummy=0)	
	DECLARE %{		DECLARE %{	
	<pre>double constant=2;</pre>	× >	double constant;	
	<pre>double two_x_dummy; double flag;</pre>		<pre>double two_x_dummy; \$ }</pre>	
th	<pre>&amp;}</pre>	•	• • 1	
ui	01		USERVARS %{	
	INITIALIZE %{		double flag;	
	<pre>two_x_dummy=2*dummy;</pre>		8}	
	8}			
			INITIALIZE %{	
	TRACE		<pre>constant=2; tuo u dummu=2tdummus</pre>	
	COMPONENT arm = Arm()		<pre>two_x_dummy=2*dummy; %}</pre>	
	AT (0, 0, 0) ABSOLUTE		01	
	EXTEND %{		TRACE	
	<pre>flag=0;</pre>			
	8}		COMPONENT arm = Arm()	
			AT (0, 0, 0) ABSOLUTE	
	COMPONENT source = Source_simple(		EXTEND %{ flag=0;	
	radius = 0.02, dist = 3,		11ag=0; %}	
	focus $xw = 0.01$ ,		01	
	focus $yh = 0.01$ ,		COMPONENT source = Source simple(	
	lambda0 = 6.0,		radius = 0.02,	
	dlambda = 0.05,		dist = 3,	
	flux = 1e8)		$focus_xw = 0.01,$	
	AT (0, 0, 0) RELATIVE arm		$focus_yh = 0.01,$ lambda0 = 6.0,	
	COMPONENT coll2 = Slit(		dlambda = 0.05,	
	radius = $0.01$ )		flux = 1e8)	
	AT (0, 0, 6) RELATIVE arm		AT (0, 0, 0) RELATIVE arm	
	EXTEND %{			
	flag=SCATTERED;		COMPONENT coll2 = Slit(	
	8}		radius = 0.01)	
	COMPONENT detector = PSD_monitor(		AT (0, 0, 6) RELATIVE arm EXTEND %{	
	nx = 128,		flag=SCATTERED;	
	ny = 128,		<pre>%}</pre>	
	filename = "PSD.dat",			
	xmin = -0.1,		COMPONENT detector = PSD_monitor(	
	xmax = 0.1		nx = 128,	
	ymin = -0.1, ymax = 0.1)		ny = 128, filename = "PSD.dat",	
	Y = 0.1 AT (0, 0, 9.01) RELATIVE arm		xmin = -0.1,	
			x max = 0.1,	
	END		ymin = -0.1,	
			ymax = 0.1)	
			AT (0, 0, 9.01) RELATIVE arm	
Peter W			PND	
	4		END	





All the stuff usually put in DECLARE earlier like

- double var=3.0;
- function declarations

are actually OK to here, but not so in the components.

For symmetry we suggest initialisation in initialise only.

Becomes part of the \_particle struct, i.e. can change with each particle

DTU SPALLATION SOURCE

# Arm unchanged

🦐 Undo 🎻 🕎 🕘 • Save [v2.5comps]...] Arm.comp P Ŀ 出 出 Arm.comp Arm.comp Files are identical Files are identical Hide Hide -McStas, neutron ray-tracing package \* McStas, neutron ray-tracing package Copyright 1997-2002, All rights reserved Copyright 1997-2002, All rights reserved Risoe National Laboratory, Roskilde, Denmark Risoe National Laboratory, Roskilde, Denmark Institut Laue Langevin, Grenoble, France Institut Laue Langevin, Grenoble, France \* Component: Arm \* Component: Arm \* %I \* %I \* Written by: Kim Lefmann and Kristian Nielsen \* Written by: Kim Lefmann and Kristian Nielsen \* Date: September 1997 \* Date: September 1997 \* Version: \$Revision\$ \* Version: \$Revision\$ \* Release: McStas 1.6 \* Release: McStas 1.6 \* Origin: Risoe \* Origin: Risoe \* Arm/optical bench \* Arm/optical bench \* %D \* %D \* An arm does not actually do anything, it is just there to set \* An arm does not actually do anything, it is just there to set \* up a new coordinate system. \* up a new coordinate system. \* %P \* %P \* Input parameters: \* Input parameters: \* %E \* %E DEFINE COMPONENT Arm DEFINE COMPONENT Arm DEFINITION PARAMETERS () DEFINITION PARAMETERS () SETTING PARAMETERS () SETTING PARAMETERS () OUTPUT PARAMETERS () OUTPUT PARAMETERS () /\* Neutron parameters: (x,y,z,vx,vy,vz,t,sx,sy,sz,p) \*/ /\* Neutron parameters: (x,y,z,vx,vy,vz,t,sx,sy,sz,p) \*/ TRACE TRACE 8{ 8{ 8} 8} MCDISPLAY MCDISPLAY €{ /\* A bit ugly; hard-coded dimensions. \*/ /\* A bit ugly; hard-coded dimensions. \*/ line(0,0,0,0.2,0,0); line(0,0,0,0.2,0,0); line(0,0,0,0,0.2,0); line(0,0,0,0,0.2,0); line(0,0,0,0,0,0.2); line(0,0,0,0,0,0.2); 8} 8} END END

[v2.5comps] Arm.comp — [v3.0comps] Arm.comp

Unicode (UTF-8) 🔻 🔹 Ln 1, Col 1 🔻



#### Source\_simple minor changes

* Origin: Risoe	* Origin: Risoe
* A circular neutron source with flat energy spectrum and arbitrary flux	* * A circular neutron source with flat energy spectrum and arbitra
<pre>* * * %D * The routine is a circular neutron source, which aims at a square target * centered at the beam (in order to improve MC-acceptance rate). The angular * divergence is then given by the dimensions of the target. * The neutron energy is uniformly distributed between lambda0-dlambda and * lambda0+dlambda or between E0-dE and E0+dE. * The flux unit is specified in n/cm2/s/st/energy unit (meV or Angs). * The statement of the statement</pre>	<pre>* * * %D * The routine is a circular neutron source, which aims at a square * centered at the beam (in order to improve MC-acceptance rate). * divergence is then given by the dimensions of the target. * The neutron energy is uniformly distributed between lambda0-dlam * lambda0+dlambda or between E0-dE and E0+dE. * The flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit is specified in n/cm2/s/st/energy unit (meV or And * the flux unit unit unit unit unit unit unit unit</pre>
* This component replaces Source_flat, Source_flat_lambda, * Source_flux and Source_flux_lambda.	* This component replaces Source_flat, Source_flat_lambda, * Source_flux and Source_flux_lambda.
<pre>* Example: Source_simple(radius=0.1, dist=2, focus_xw=.1, focus_yh=.1, E0=14, dE=2) *</pre>	* * Example: Source_simple(radius=0.1, dist=2, focus_xw=.1, focus_y) *
<pre>* %P * radius: [m] Radius of circle in (x,y,0) plane where neutrons * yheight: [m] Height of rectangle in (x,y,0) plane where neutron * xwidth: [m] Height of rectangle in (x,y,0) plane where neutron * target_index: [1] relative index of component to focus at, e.g. ne: * dist: [m] Distance to target along z axis. * focus_xw: [m] Width of target * focus_yh: [m] Height of target * E0: [meV] Mean energy of neutrons. * dE: [meV] Energy half spread of neutrons. * dlambda: [AA] Wavelength half spread of neutrons. * flux: [1/(s*cm**2*st*energy unit)] flux per energy unit, Angs or meV if flux=0, the</pre>	<pre>* %P * radius: [m] Radius of circle in (x,y,0) * yheight: [m] Height of rectangle in (x,y) * xwidth: [m] Width of rectangle in (x,y,4) * target_index: [1] relative index of component * dist: [m] Distance to target along z a * focus_xw: [m] Width of target * focus_w: [m] Height of target * focus_yh: [m] Height of target * E0: [meV] Mean energy of neutrons. * dE: [meV] Energy half spread of neutrons * dlambda: [AA] Wavelength half spread of neutrons * flux: [1/(s*cm**2*st*energy unit)] flux per energy unit, Angs</pre>
<pre>* gauss: [1] Gaussian (1) or Flat (0) energy/wavelength distr. * %E **********************************</pre>	<pre>* gauss: [1] Gaussian (1) or Flat (0) end * * %E ********************************</pre>
DEFINE COMPONENT Source_simple DEFINITION PARAMETERS () SETTING PARAMETERS (radius=0.1, yheight=0, xwidth=0, dist=0, focus_xw=.045, focus_yh=.12, E0=0, dE=0, lambda0=0, dlambda=0,	DEFINE COMPONENT Source_simple DEFINITION PARAMETERS () SETTING PARAMETERS (radius=0.1, yheight=0, xwidth=0, dist=0, focus_xw=.045, focus_yh=.12, E0=0, dE=0, lambda0=0, dlambda=0,
flux=1, gauss=0, int target_index=+1)  DUTPUT PARAMETERS (pmul,square,srcArea)	<pre>flux=1, gauss=0, int target_index=+1) X OUTPUT PARAMETERS (pmul,square,srcArea,tx,ty,tz)</pre>
<pre>/* Neutron parameters: (x,y,z,vx,vy,vz,t,sx,sy,sz,p) */ DECLARE</pre>	<pre>/* Neutron parameters: (x,y,z,vx,vy,vz,t,sx,sy,sz,p) */ DECLARE</pre>
<pre>%{ double pmul, srcArea; X</pre>	%{ ★ double pmul; ←
int square;	double srcArea;
double tx,ty,tz;	int square;
8	X double tx; ←
INITIALIZE	double ty;
square = 0;	*}
/* Determine source area */	INITIALIZE
if (radius && !yheight && !xwidth ) {	8{
square = 0;	square = 0;
<pre>srcArea = PI*radius*radius;</pre>	/* Determine source area */
<pre>} else if(yheight &amp;&amp; xwidth) {</pre>	if (radius && !yheight && !xwidth ) {
<pre>square = 1; srcArea = xwidth * yheight;</pre>	square = 0; srcArea = PI*radius*radius;
}	} else if(yheight && xwidth) {
	square = 1;
if (flux) {	<pre>srcArea = xwidth * yheight;</pre>
<pre>pmul=flux*le4*srcArea/mcget_ncount(); if (dlorb de strcArea/mcget_ncount();</pre>	}
if (dlambda) pmul *= 2*dlambda:	if (flux) 4
Unicode (UTF-8) 🔻 👗 Ln 56, Col 1 👻	Unicode (UT



# Slit unchanged

🖻 🔛 Save 🥱 Undo 🇬 🥎 🧶 🧶	
[v2.5compss] Slit.comp ×	
🛓 📄 Slit.comp	L Slit.comp
Files are identical     Hide	Files are identical
<pre>Kottas, neutron ray-tracing package Kottas, neutron ray-tracing package Kottason kained Kottason ka</pre>	<pre>Version Section S</pre>
<pre>%} TRACE %{     PROP_Z0;     if (((radius == 0) &amp;&amp; (x<xmin x=""   ="">xmax    y<ymin y=""   ="">ymax))        ((radius != 0) &amp;&amp; (x*x + y*y &gt; radius*radius))) </ymin></xmin></pre>	<pre>%} TRACE %{     PROP_Z0;     if (((radius == 0) &amp;&amp; (x<xmin x=""   ="">xmax    y<ymin y=""   ="">ymax))       ((radius != 0) &amp;&amp; (x*x + y*y &gt; radius*radius))) </ymin></xmin></pre>
Unicode (UTF-8) 🔻 🔻 Ln 104, Col 4 🔻	Unicode (UTF-8) 🔻

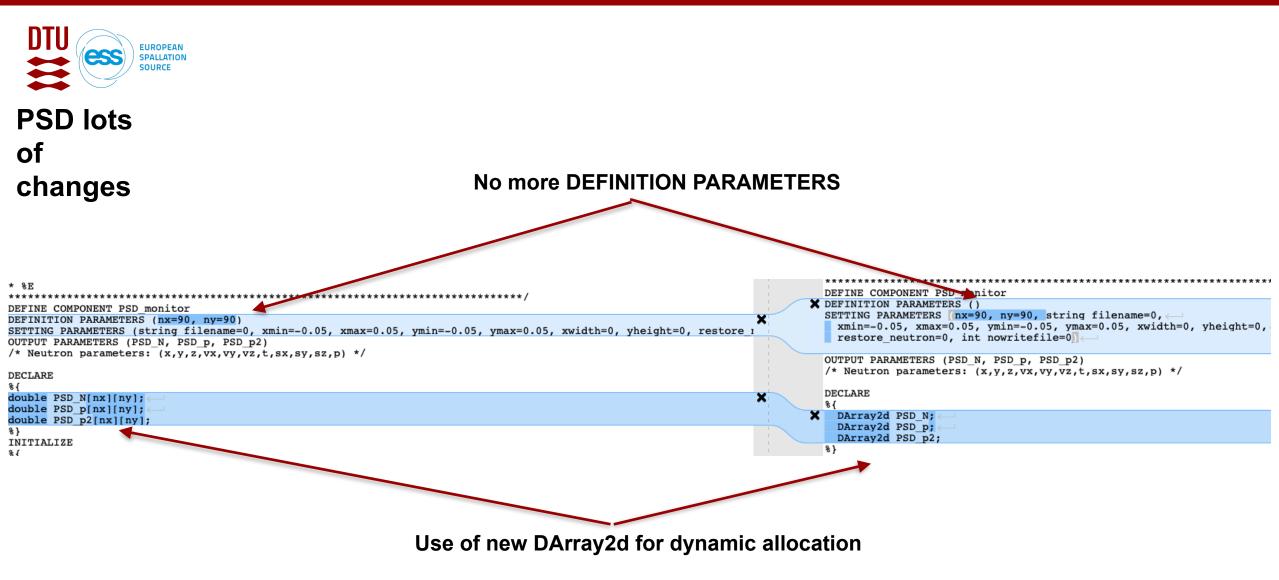
[v2.5comps] Slit.comp — [v3.0comps] Slit.comp



#### PSD lots of changes

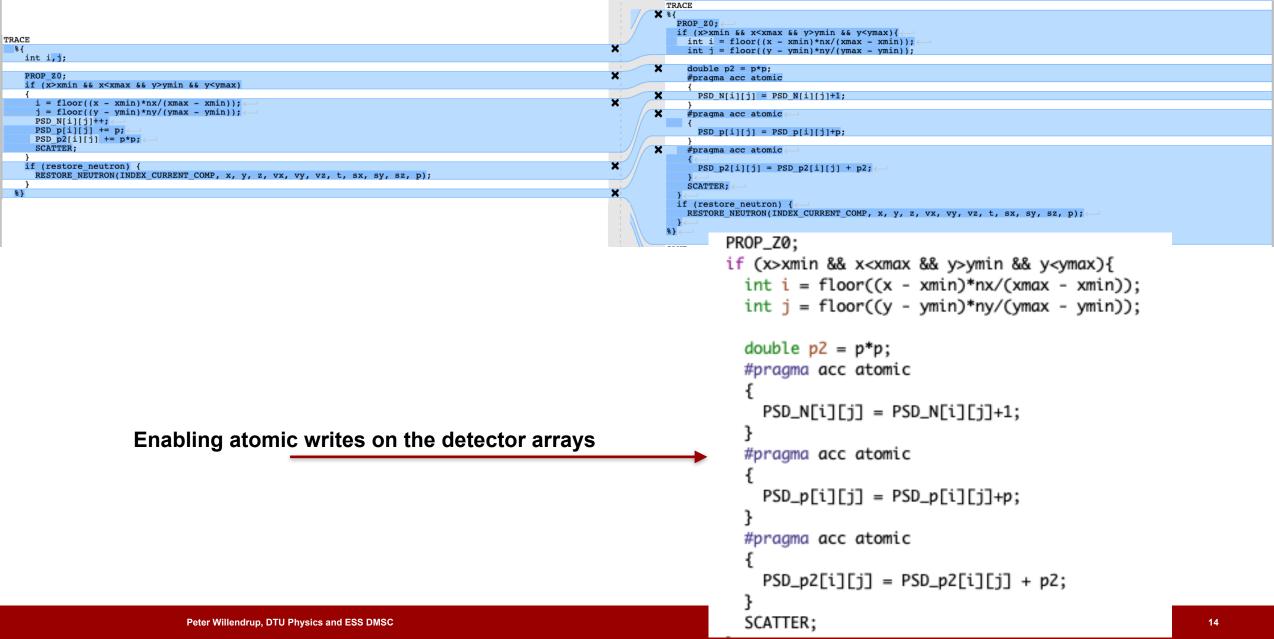
PSD_monitor.comp		B PSD_monitor.comp
ilename: [string]       Name of file in which to store the detector image         estore_neutron: [1]       If set, the monitor does not influence the neutron state         owritefile: [1]       If set, monitor will skip writing to disk         UTPUT PARAMETERS:         SD N: []       Array of neutron counts		<pre>bb bFFINE COMPONENT PSD monitor bFFINE COMPONENT PSD monitor bFFINITION PARAMETERS () SETTING PARAMETERS inx=90, ny=90, string filename=0,</pre>
SD_p: [] Array of neutron weight counts SD_p2: [] Array of second moments E The COMPONENT PSD monitor		OUTPUT PARAMETERS (PSD_N, PSD_P, PSD_P2) /* Neutron parameters: (x,y,z,vx,vy,vz,t,sx,sy,sz,p) */ DECLARE %{
INITION PARAMETERS (nx=90, ny=90) TING PARAMETERS (string filename=0, xmin=-0.05, xmax=0.05, ymin=-0.05, ymax=0.05, xwidth=0, yheight=0, y PUT PARAMETERS (PSD N, PSD_p, PSD_p2) Neutron parameters: (x,y,z,vx,vy,vz,t,sx,sy,sz,p) */	restore 1	<pre>X DArray2d PSD_N; DArray2d PSD_p; DArray2d PSD_p; DArray2d PSD_p2; %} INITIALIZE %{</pre>
LARE		<pre>if (xwidth &gt; 0) { xmax = xwidth/2; xmin = -xmax; } if (yheight &gt; 0) { ymax = yheight/2; ymin = -ymax; }</pre>
ble PSD_N[nx][ny];	×	<pre>X if ((xmin &gt;= xmax)    (ymin &gt;= ymax)){     printf("PSD monitor: %s: Null detection area !\n"     "ERROR (xwidth,yheight,xmin,xmax,ymin,ymax). Exiting",     NAME_CURRENT_COMP);     exit(0);</pre>
i, j;	×	}
<pre>(xwidth &gt; 0) { xmax = xwidth/2; xmin = -xmax; } if (yheight &gt; 0) { ymax = yheight/2; ymin = -ymax; } if ((xmin &gt;= xmax)    (ymin &gt;= ymax)) {</pre>	×	<pre>X PSD_N = create_darr2d(nx, ny); PSD_p = create_darr2d(nx, ny); PSD_p2 = create_darr2d(nx, ny);</pre>
<pre>printf("PSD monitor: %s: Null detection area !\n" "ERROR (xwidth,yheight,xmin,xmax,ymin,ymax). Exiting",</pre>		<pre>X int i, j;</pre>
<pre>for (i=0; i<nx; (j="0;" <="" for="" i++)="" j++)="" j<ny;="" pre="" psd_n(i](j)="0;" {=""></nx;></pre>	×	<pre>PSD_p2[i][j] = 0;</pre>
PSD_p(i)[j] = 0; PSD_p2(i)[j] = 0; }	×	TRACE %{ PROP_Z0;
CE { int i,j;	×	<pre>if (x&gt;xmin &amp;&amp; x<xmax &&="" y="">ymin &amp;&amp; y<ymax){ (xmax="" (ymax="" -="" i="floor((x" int="" j="floor((y" pre="" xmin));="" xmin)*nx="" ymin));<="" ymin)*ny=""></ymax){></xmax></pre>
PROP_Z0; if (x>xmin && x <xmax &&="" y="">ymin &amp;&amp; y<ymax)< td=""><td>×</td><td><pre>double p2 = p*p; #pragma acc atomic</pre></td></ymax)<></xmax>	×	<pre>double p2 = p*p; #pragma acc atomic</pre>
<pre>{     i = floor((x - xmin)*nx/(xmax - xmin));     j = floor((y - ymin)*ny/(ymax - ymin));     PSD_N[i][j]+;     PSD_P[i][i]] + ;     PSD_P[i][i]] + ;     p;     p;</pre>	×	<pre>X { PSD_N[i][j] = PSD_N[i][j]+1;</pre>
PSD_p2(i)(j) += p*p; SCATTER;		} ★ #pragma acc atomic ←
) if (restore_neutron) { RESTORE NEUTRON(INDEX_CURRENT_COMP, x, y, z, vx, vy, vz, t, sx, sy, sz, p); }	×	<pre>PSD_p2[i](j] = PSD_p2[i](j] + p2;</pre>
} E {	××	if (restore neutron) {
<pre>if (Inowritefile) {     DETECTOR_OUT_2D(         "PSD monitor",         "X position [cm]",         "Y position [cm]",</pre>		RÉSTORE_NEUTRON(INDÈX_CURRENT_COMP, x, y, z, vx, vy, vz, t, sx, sy, sz, p); } } SAVE
<pre>xmin*100.0, xmax*100.0, ymin*100.0, ymax*100.0, nx, ny, &amp;PSD_N[0][0],&amp;PSD_p[0][0],&amp;PSD_p2[0][0], filename); }</pre>		<pre>if (!nowritefile) {     DETECTOR_OUT_2D;     "PSD monitor",     "X position [cm]",     "Y position [cm]",     min*100.0, mmax*100.0, ymax*100.0,</pre>

[v2.5comps] PSD\_monitor.comp\* — [v3.0comps] PSD\_monitor.comp





#### **PSD** lots of changes





#### **Generated code: LOTS of changes**

<pre>c Comple: cc -&gt; Minimal.out example v25.c cFLAGS= // / / / / / / / / / / / / / / / / /</pre>	example_v2mple_v30.c ×		
<pre></pre>		JD.	
/ Formation         APPLIE Source and the standard strate we have a standard strate we hav			
Context of the second s	* Automatically generated file. Do not edit.		/* Automatically generated file. Do not edit.
Internation response of large (states)         X			
<pre>base pris. pris. pris. pris. pris. 200 '</pre>	<pre>Cleator: Acstas <a href="http://www.mcstas.org/">http://www.mcstas.org/</a> # Instrument: avample u25 instr (Winimal)</pre>	¥.	* Instrument, example use instructions of y
<pre>itig: cemptering ifig: cemptering //dx/ ////dx/ ////dx/ //dx/ /d</pre>	Date: Fri Jan 17 14:24:26 2020		
<pre>compage to the Minimula sample voice // def // def //</pre>	* File: example v25.c		
<pre>//***********************************</pre>	Compile: cc -o Minimal.out example_v25.c		* CFLAGS=
<pre>che wood Fills Wood Fills () Wets 3.0-der / Au (), 200" define Wood Wets 3.0-der / Au (), 2</pre>	CFLAGS=		*/
<pre>see according filling "belies 2.3 - Osc. 12, 2014"</pre>	/		★ #define MCCODE STRING "McStas 3.0-dev - Jan. 17. 2020"
<pre>still the Theory Text and the second se</pre>			#define FLAVOR "mcstas"
<pre>dise Express Monore Monor</pre>		×	#define FLAVOR_UPPER "MCSTAS"
<pre>stics will be used of the second of the</pre>	eine FLAVOR "mcStas"		Hadding NO HOE DEBNING NATH
<pre>stds #_Descriptions if is model to set the standard is the similation program. Here we have defined with the isst composed in the similation program. Here we have is the standard is the sis the</pre>	eile Flavor_UPPER MCSTAS		#define MC_USE_DEFAUDT_MAIN
<pre>define VF *** ********************************</pre>			
<pre>set 1 = monote - set intermediate - set of the set</pre>			
<pre>Added # 2.27 Added # 2.27</pre>	ine 1 "mccode-r.h"	×	
<pre>Medica : 0.0000 =</pre>	*************		#ifndef M 2 PI
<pre>Medds _ metters/rev revired/spreader.h Heids Heids Libertown, beside, memory Initiation Libertown, beside, memory Initiation, beside, for Methal/Metricase. Initiation Libertown, beside, memory Initiation, beside, for Methal/Metricase. Initiation Libertown, beside, memory Initiation, beside, for Methal/Metricase. Initiation Libertown, beside, memory Initiation, beside, for Methal/Metricase. Initiation, beside, for Methal/Metri</pre>			#define M_2 PI 0.63661977236758134308
Corright (0, 197-200, All rights reserved Institute (as Lawyork, Geobbs, Fasce         Institute law Lawyork, Geobbs, Fasce         Munities is hard/accode-1.h	McCode, neutron/xray ray-tracing package		#endif
Institut Lave Lawysis, Greenble, France  Institut Lave Lawysis, Greenble, France  Institut Lave Lawysis, Greenble, France  Institut Auge Lawysis, Greenble, Fran	Copyright (C) 1997-2009, All rights reserved		#ifndef M PI 2
<pre>harding share/secode-r.h these share/sec</pre>	Risoe National Laboratory, Roskilde, Denmark		
Mantine share/meeder.h Weaking testing and the share/meeder.h Weaking testing and the share/meeder.h Weaking testing and the share of the Media/Mediane weaking Weaking testing and the share of the Media/Mediane weaking Weaking testing and the share of the Stare weaking the share weaking Weaking testing and the share weaking the	Institut Laue Langevin, Grenoble, France		
<pre>statestication statestication the desired (see details in the code) first was 37, 197 first was 37, 197 first was 37, 197 first was an external liberry, the following variables and macros must be desired (see details in the code) first was an external liberry, the following variables and macros must be desired (see details in the code) first was an external liberry, the following variables and macros must be desired (see details in the code) first was an external liberry, the following variables and macros must be desired (see details in the code) first was an external liberry, the following variables and macros must be desired (see details in the code) first was an external liberry, the following variables and macros must be desired (see details in the code) first was an external liberry, the following variables attent MCWM accore stored 1;     double way, was 1 / the desired (see details on the code) first was an external liberry, the following variables attent MCWM accore stored 1;     double way, was 1 / the desired (see details on the code) first was an external liberry, the following variables first WCWM accore stored 1;     double way, was 1 / the desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the code) first was an external liberry desired (see details on the see detail</pre>			
<pre>bide.ticetion bide.ticetion bide.ticeti</pre>	Runtime: share/mccode-r.h		#detine M_SQRT2 1.4142135623730951 /* sqrt(2) */
<pre>kritten by: RS writes var all by: RS writes var address for Kolta/Koltrace. In order to use this likery as a strent kingertalise and accor mut be delated (see detail in the code) finite writes in the delated (see detail in the code) finite writes in the delated (see detail in the simulation program, see definitions can be dead statis. */ forder Kong Randhames var; class particle and because (see variables finite writes in the simulation program, see definitions can be dead statis. */ forder Kong Randhames var; class particle and because (see variables finite writes in the simulation program, see definitions can be dead statis. */ forder Kong Randhames var; class particle and because (see variables finite writes in the simulation program, see definitions can be dead statis. */ forder Kong Randhames var; class particle is abodied in the simulation program, see definitions can be dead statis. */ forder Kong Randhames var; class particle is abodied in the simulation program, see definitions can be dead statis. */ forder Kong Randhames var; class particle is particle global randhames var; class particle is particle is particle is particle is part</pre>			#endlr
<pre>backs may 29, 197 Wancie system header for Kotkas/Kottrace. Manutie system header for Kotkas/Kottrace. In order to use this library as an external library, the following variables and macros must be decircle (see extails in the code) attract moinputable struct particle { function Kottas/Kottrace. For the contexp decircle (see extails in the code) attract moinputable struct particle { function Kottas/Kottrace. Adding function Kottas/Kottrace. For Kottas/Kottrace. For Kottas/Kottrace. For Contexp decircle (see extails in the code) attract moinputable struct particle { function Kottas/Kottrace. For Kottas/Kottrac</pre>			
<pre>Metas 2.5 Metas 2.5 M</pre>	Written by: KN		#IIDEF DISABLE TRACE
Version Skevision5 Multise system heads for KStas/KStrace. In order to use this library as an external library, the following variables find decrease and the declared (see declared (se	Date: Aug 23, 1997	•	
<pre>kartine system header for KoStas/KcXtrace. in order to use this library as an external library, the following variables and macro maints be declared (as the code) artoch meingstable struct moinpstable(); artoch meingstable struct moinpstable struct moin</pre>			#cluit
<pre>Munifies system header for MoStas/MoStraee. In order to use this libery as an external libery, the following variables and sacross must be declared (see details in the code) first methylicitable struct information (see details in the code) intermention (see details (see details (see details event */ intermention (see details (see details (see details event */ intermention (see details (see details (see details (see details event */ intermention (see details (see deta</pre>	version. vevisions		#ifdef USE PGT
<pre>in order to use this library as an external library, the following variables and marrow must be decired (see details in the code) attract mcinputable struct mcinputable (); there decired (see details in the code) attract mcinputable struct mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (); there decired (see details in the code) attract mcinputable (see details in the code) attract mcinputable (); there decired (see details (see deta</pre>	Runtime system header for McStas/McXtrace.		
In order to use this library as an external library, the following variables and ancom must be declared (pred details in the code) struct mcinputable struct mcinputable struct mcinputable struct mcinputable struct mcinputable modef antenna struct mcinputable struct mcinputable modef antenna struct mcinputable struct mcinputable modef antenna structure mcinputable mod			#include <openacc curand.h=""></openacc>
and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and marco must be declared (see details in the code) #truct minutes and be added in the simulation program, see definitions can be adde static #truct minutes and be added for the simulation program, see definitions can be adde static #truct minutes and marco marco must be readed (see definitions can be added static definitions can be added static definitions can be adde static definitions can be added in the simulation program, see definitions can be added static for works and marco marc	In order to use this library as an external library, the following variables		
<pre>struct_stru</pre>	and macros must be declared (see details in the code)		
<pre>int monumipar; int monumipar; char meintained: name() meintain; extern KUNN acceptore(); extern KUNN acceptore(); ex</pre>			
<pre>int stracemabled, modefaultain; overam KUNN mobile for Kestas/Kestrace version* Usage: Automatically embeded in the c code. 51d5 51d5 51d6 51d6 KCODE R H efiles K</pre>	<pre>struct mcinputtable_struct mcinputtable[];</pre>		double x,y,z; /* position [m] */
<pre>int Extracemabled, mode subscription is the second of the second of</pre>		×	double vx,vy,vz; /* velocity [m/s] */
<pre>extern MXNN mccomp_storein[]; extern MXNN mccomp_storein[]; e</pre>	char mcinstrument_name[], mcinstrument_source[];		double sx,sy,sz; /* spin [0-1] */
<pre>extern KLNUM mcseattered; define mcCoup_STRIM STM mcseattered; define mcCoup_STRIM STM mcseattered; the McStas/McXtrace version" Usage: Automatically embeded in the c code. findef MCCOUP_R.H finde MCCOUP_</pre>	int mctraceenabled, mcdefaultmain;		unsigned long randstate //;
<pre>extern KLNW mckattered; define McCODE_STRIK "In McStas/KcXtrace version" define McCODE_STRIK "In McStas/KcXtrace version" define mcCode_Strik "A define McStas/KcXtrace version" define mcCode_Strik "In McStas/KcXtrace version" define mcCode_Strik "In McStas/KcXtrace version" define mcCode_Strik "In McStas/KcXtrace. inded string.b nclude string.b stri</pre>	extern MCNOM mcCompstorell[];		long with the upper to the
<pre>#define MCCODE_STRING "the McStas/McXtrace version" Usage: Automatically embeded in the c code. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatically embeded in the simulation program, some definitions can be and static. Usage: Automatical embedded in the simulation program, some definitions can be and static. Usage: Automatical embedded in the simulation program. Usage: Automatical embedded in the simulation program. Usage: Automatical embedded in the simulat</pre>			long long very /* event if any where to send this event */
<pre>Gase: Automatically embbeded in the c code. Sids Sids ************************************</pre>	#define MCCODE STRING "the McStas/McStrace version"		longindex, /* component index when this event is to be removed/ignored */
Usage: Automatically embleded in the c code. StdS interventional state of the c code. StdS interventional state of the c code. Interventional state of the code. Intervention state of the code.			long scattered; / flag set to TRUE when this event has interacted with the last component instance */
StdS       // user variables:         fndef MCCODE R H       (lass particle class particle;         fndef MCCODE R H       (lass particle particle class particle;         fndef MCCODE R H       (lass particle particle class particle;         include (math.b)       (lass particle particle class particle;         nolude (math.b)       (lass particle particle class particle;         nolude (stdlib.b)       (lass particle particle class particle;         nolude (stdlib.b)       (lass particle particle class particle;         nolude (stdlib.b)       (lass particle)         nolude (stdlib.b)       (lass particl	Usage: Automatically embbeded in the c code.		long restore: /* set to true if neutron event must be restored */
Sids       double flag;         ************************************			// user variables:
<pre>// // // // // // // // // // // // //</pre>	\$Id\$		
fndef MCCODE R.H         efine MCCODE R.H         class particle particle global_randhbuse var;         class particle mccoder.h         nclude statio.h>         nclude static.*         for MC EMBRODED RUNTIME         efine mestatic static         ndif         fdef dest os             fide f dest os           if de dest os            if de dest os                    in order to use this library as an external library, the following variables			};
<pre>efine MCCODE_R_H "\$Revisions" nelude <math.h> nelude <math.h> nelude <math.h> nelude <string.h> n</string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></math.h></math.h></math.h></pre>	***************************************		typedef struct _struct _particle _class_particle;
<pre>lefine MCCODE_R_H "\$Revisions" nelude <math.h> nelude <math.h< td=""></math.h<></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></math.h></pre>	fndef MCCODE P H		class particle particle global randobuse war.
<pre>nclude <math.h> nclude <string.h> nclude <s< td=""><td></td><td></td><td>class particle* particle = &amp; particle global randnbuse var;</td></s<></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></math.h></pre>			class particle* particle = & particle global randnbuse var;
<pre>clude <string.h> clude <string.h> c</string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></string.h></pre>			
<pre>hclude <stdii.h> hclude <stdii.h <stdii.h="" <stdiih="" hc<="" hclude="" td=""><td></td><td>×</td><td></td></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></stdii.h></pre>		×	
helude <stdio.h> helude <stdio.h> helude <limits.h> helude <limits.h> helude <rno.h> helude <rno.h> helude <fioat.h> helude <fioat.h> helude <fioat.h> helude <fioat.h> helude <fioat.h> helude <inttypes.h> If the runtime is embedded in the simulation program, some definitions can be made static. */ fidef MC EMBEDDED RUNTIME file mestatic static lse effine mestatic fidef dest os fidef dest os</inttypes.h></fioat.h></fioat.h></fioat.h></fioat.h></fioat.h></rno.h></rno.h></limits.h></limits.h></stdio.h></stdio.h>	nclude <string.h></string.h>		<pre>X /* embedding file "mccode-r.h" */</pre>
naluda <stadara,h> naluda <stadara,h> natuta Laue Laboratory, Roskilde, Denmark nastiut Laue Laboratory, Roskilde, Denmark * Runtime: share/mccode-r.h * Stidentification * Virticen by: KN * Versions * Versions * Versions * Runtime system header for McStas/McXtrace. * In order to use this library as an external library, the following variables</stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h></stadara,h>			
hclude <limits.h> hclude <limits.h> hclude <limits.h> hclude <lime.h> hclude </lime.h> hclude </lime.h> hclude </lime.h> hclude </lime.h> hclude  hclude </lime.h> hclude <td></td><td></td><td>/**************************************</td></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></lime.h></limits.h></limits.h></limits.h>			/**************************************
<pre>helude <erro.h> holude <erro.h> holude <intrime.h> holude <intrim< td=""><td>IcLude Stoarg.n/</td><td></td><td>* Macada noutron (xray ray tracing pagkaga</td></intrim<></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></intrime.h></erro.h></erro.h></pre>	IcLude Stoarg.n/		* Macada noutron (xray ray tracing pagkaga
<pre>hclude <time.h> hclude <time.h> hclude <time.h> hclude <inthypes.h> hclude <inthypes.h< <inthy<="" <inthypes.h<="" hclude="" td=""><td></td><td></td><td>- metode, medition/xiay iay-tiating package</td></inthypes.h<></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></inthypes.h></time.h></time.h></time.h></pre>			- metode, medition/xiay iay-tiating package
<pre>nclude <float.b> nclude <float.b> nclude <intypes.b> If the runtime is embedded in the simulation program, some definitions can be made static. */ fdef MC_EMBEDDED_RUNTIME ffine mostatic static se ffine mostatic se ffine mostatic ffine mostatic ffine mostatic se ffine mostatic ffine mostatic ffine mostatic ffine mostatic ffine mostatic se ffine mostatic ffine mostatic</intypes.b></float.b></float.b></pre>			* Bisce National Laboratory, Reskilde, Denmark
Aclude <inttypes.h> <pre></pre></inttypes.h>			
If the runtime is embedded in the simulation program, some definitions can be made static. */ fdef MC_EMBEDDED_RUNTIME fine mcstatic static se fine mcstatic static fdef dest os fdef dest os			*
If the runtime is embedded in the simulation program, some definitions can       *         be made static. */       *         fdef MC_EMBEDDED RUNTIME       *         fdef MC_EXABLE DER RUNTIME       *         lse       *         efine mostatic       *         rdif       *         fdef dest os       *			* Runtime: share/mccode-r.h
be made static. */ fdef dest os  * Widentification * Widentification * Widentification * Writen by: KN * Date: Aug 29, 1997 * Date: Aug 29, 1997 * Version: SRevisionS * Version: SRevisionS * Version: SRevisionS * Runtime system header for McStas/McXtrace. * In order to use this library as an external library, the following variables	If the runtime is embedded in the simulation program, some definitions can		*
fdef MC EMBEDDED RUNTIME       * Date: Aug 29, 1997         lefine mcstatic       * Release: McStase: McStase: McStas/McXtrace.         lefe       * Version: SRevisionS         refine mcstatic       * Runtime system header for McStas/McXtrace.         // fdef       dest os	be made static. */		
lefine mcstatic static       X       * Release: McStas 3.0-dev         oblee       * Version: SkevisionS         endif       *         indef dest os       * In order to use this library as an external library, the following variables			
lse * Version: \$Revision\$ efine mcstatic * * Runtime system header for McStas/McXtrace. fdef dest os * In order to use this library as an external library, the following variables			* Date: Aug 29, 1997
efine mcstatic       *         ndif       *         fdef       dest os         *       In order to use this library as an external library, the following variables		×.)	
hdif fdef dest os * In order to use this library as an external library, the following variables			* Version: \$Revision\$
fdef dest os * * In order to use this library as an external library, the following variables	elle mostatic	×	* ·
fdefdest_os * In order to use this library as an external library, the following variables	Jair		<pre>* kuntime system header for McStas/McXtrace.</pre>
* In order to use this library as an external library, the following variables	fdaf dagt og		* * The order to use this likeway as an external likeway the following severables
			· In order to use this library as an external inprary, the following variables

Unicode (UTF-8) V C V Ln 4, Col 1 V



#### The neutron and "state-flags" in the instrument

v2.5: Global variables

double x, y, z, vx, vy, vz, t, sx, sy, sz, p; double flag;

```
v3.0: particle struct, including any USERVARS like flag.
struct _struct_particle {
  double x,y,z; /* position [m] */
  double vx,vy,vz; /* velocity [m/s] */
  double sx,sy,sz; /* spin [0-1] */
 unsigned long randstate[7];
  double t, p; /* time, event weight */
  long long _uid: /* event ID */
  long _index; /* component index where to send this event */
  long _absorbed; /* flag set to TRUE when this event is to be removed/ignored */
  long _scattered; /* flag set to TRUE when this event has interacted with the last component instance */
  long _restore: /* set to true if neutron event must be restored */
 // user variables:
 double flag;
};
typedef struct _struct_particle _class_particle;
```



### Input parameters and the instrument struct

```
struct _struct_instrument_parameters {
 MCNUM _dummy;
};
typedef struct _struct_instrument_parameters _class_instrument_parameters:
struct _instrument_struct {
  char __name[256]; /* the name of this instrument e.g. 'Minimal' */
/* Counters per component instance */
  double counter_AbsorbProp[6]; /* absorbed events in PROP routines */
  double counter_N[6], counter_P[6], counter_P2[6]; /* event counters after each component instance */
  _class_particle _trajectory[6]; /* current trajectory for STORE/RESTORE */
/* Components position table (absolute and relative coords) */
 Coords _position_relative[6]; /* positions of all components */
 Coords _position_absolute[6];
  _class_instrument_parameters _parameters; /* instrument parameters */
} _instrument_var;
struct _instrument_struct *instrument = & _instrument_var;
```

... in e.g. an EXTEND section one should now use the macro INSTRUMENT\_GETPAR(dummy) which translates the bare dummy into instrument->\_parameters.\_dummy

```
#define INSTRUMENT_GETPAR(par) (instrument->_parameters._ ## par)
```



ł

}

#### **Declare section**

/\* User declarations from instrument definition. Can define functions. \*/
 double constant;
 double two\_x\_dummy;

# **Initialise section**

class\_PSD\_monitor\_init(&\_detector\_var);

```
#define dummy (instrument->_parameters._dummy)
constant=2:
two_x_dummy=2*dummy;
#undef dummy
_arm_setpos(); /* type Arm */
_source_setpos(); /* type Source_simple */
_coll2_setpos(); /* type Slit */
_detector_setpos(); /* type PSD_monitor */
/* call iteratively all components INITIALISE */
class_Source_simple_init(&_source_var);
class_Slit_init(&_coll2_var);
```

Functions per component with related component structs

```
/* Parameters for component type 'PSD_monitor' */
              EUROPEAN
              SPALLATION
                                                                            struct _struct_PSD_monitor {
              SOURCE
                     Component struct example
                                                                                       _name[256]; /* e.g. detector */
                                                                              char
                                                                                       _type[256]; /* PSD_monitor */
                                                                              char
                                                                                       _index; /* e.g. 2 index in TRACE list */
                                                                              long
/* component detector=PSD_monitor() [4] DECLARE */
                                                                              Coords _position_absolute;
/* Parameter definition for component type 'PSD_monitor' */
                                                                              Coords _position_relative; /* wrt PREVIOUS */
struct _struct_PSD_monitor_parameters {
                                                                              Rotation _rotation_absolute:
 /* Component type 'PSD_monitor' setting parameters */
                                                                              Rotation _rotation_relative; /* wrt PREVIOUS */
 MCNUM nx;
                                                                                       _rotation_is_identity:
                                                                              int
 MCNUM ny;
                                                                              _class_PSD_monitor_parameters _parameters;
 char filename[16384];
                                                                            };
 MCNUM xmin;
                                                                            typedef struct_PSD_monitor _class_PSD_monitor;
 MCNUM xmax;
                                                                            _class_PSD_monitor _detector_var;
                             Built from component definition
 MCNUM ymin;
                                                                            #pragma acc declare create ( _detector_var )
 MCNUM ymax;
 MCNUM xwidth:
 MCNUM yheight;
 MCNUM restore_neutron;
 long nowritefile:
 /* Component type 'PSD_monitor' private parameters */
 /* Component type 'PSD_monitor' DECLARE code stored as structure members */
 DArray2d PSD_N;
 DArray2d PSD_p;
 DArray2d PSD_p2;
}; /* _struct_PSD_monitor_parameters */
                                                                                       OpenACC clause to define
typedef struct _struct_PSD_monitor_parameters _class_PSD_monitor_parameters;
                                                                                       variable in device scope.
```



## Component init function example

#define radius (\_comp->\_parameters.radius) #define yheight (\_comp->\_parameters.yheight) #define xwidth (\_comp->\_parameters.xwidth) #define dist (\_comp->\_parameters.dist) #define focus\_xw (\_comp->\_parameters.focus\_xw) #define focus\_yh (\_comp->\_parameters.focus\_yh) #define E0 (\_comp->\_parameters.E0) #define dE (\_comp->\_parameters.dE) #define lambda0 (\_comp->\_parameters.lambda0) #define dlambda (\_comp->\_parameters.dlambda) #define flux (\_comp->\_parameters.flux) #define gauss (\_comp->\_parameters.gauss) #define target\_index (\_comp->\_parameters.target\_index) #define pmul (\_comp->\_parameters.pmul) #define square (\_comp->\_parameters.square) #define srcArea (\_comp->\_parameters.srcArea) #define tx (\_comp->\_parameters.tx) #define ty (\_comp->\_parameters.ty) #define tz (\_comp->\_parameters.tz) SIG\_MESSAGE("[\_source\_init] component source=Source\_simple() INITIALISE [Source\_simple.comp:68]"); square = 0;/\* Determine source area \*/ if (radius && !yheight && !xwidth ) { square = 0; srcArea = PI\*radius\*radius; } else if(yheight && xwidth) { square = 1; srcArea = xwidth \* yheight; } Contains component initialise if (flux) { pmul=flux\*1e4\*srcArea/mcget\_ncount(); section if (dlambda) pmul \*= 2\*dlambda; else if (dE) pmul \*= 2\*dE; } else { gauss = 0;pmul=1.0/(mcget\_ncount()\*4\*PI); .... etc

\_class\_Source\_simple \*class\_Source\_simple\_init( \*\_comp ) {



Instrument and component structs built on CPU and transferred to **GPU** using **OpenACC** pragmas at the end of

#### #ifdef USE PGI include <openacc.h> # acc\_attach( (void\*)&\_arm\_var ); acc\_attach( (void\*)&\_source\_var ); acc\_attach( (void\*)&\_coll2\_var ); acc\_attach( (void\*)&\_detector\_var ); #pragma acc update device(\_arm\_var) #pragma acc update device(\_source\_var) #pragma acc update device(\_coll2\_var) #pragma acc update device(\_detector\_var) acc\_attach( (void\*)&\_instrument\_var ); #pragma acc update device(\_instrument\_var) #endif

Similar "host" update in FINALLY

# INITIALISE



# Main trace Loop (v1)

```
/* loop to generate events and call raytrace() propagate them */
void raytrace_all(unsigned long long ncount, unsigned long seed) {
  /* CPU-loop */
  unsigned long long loops;
  long innerloop=2147483647;
  loops = ceil((double)ncount/innerloop);
  if (ncount>innerloop) {
    printf("Defining %llu CPU loops around kernel and adjusting ncount\n", loops);
    mcset_ncount(loops*2147483647);
  } else {
    loops=1;
    innerloop = ncount;
                                                                                     CPU may execute
                                                                                     multiple
  for (unsigned long long cloop=0; cloop<loops; cloop++) {</pre>
    if (loops>1) fprintf(stdout, "%d...", (int)cloop); fflush(stdout);
                                                                                     GPU loops
                                                               Main parallel pragma that define kernel
    #pragma acc parallel loop 🗲
    for (unsigned long pidx=0 ; pidx < innerloop ; pidx++) {</pre>
      _class_particle particleN = mcgenstate(); // initial particle
      _class_particle* _particle = &particleN;
      particleN._uid = pidx;
      long seq = pidx + seed;
      srandom(_hash(pidx + seed));
      raytrace(_particle);
                                                               Cogen-constructed function with calls
    } /* inner for */
                                                               to component trace functions
    seed = seed+innerloop;
  } /* CPU for */
  printf("\n");
  /* raytrace_all */
```



# Main trace Loop (v1)

#pragma acc routine seq

int raytrace(\_class\_particle\* \_particle) { /\* single event propagation, called by mccode\_main for Minimal:TRACE \*/ /\* init variables and counters for TRACE \*/ #undef ABSORB0 #undef ABSORB #define ABSORB0 do { DEBUG\_ABSORB(); MAGNET\_OFF; ABSORBED++; return(ABSORBED); } while(0) #define ABSORB ABSORB0 DEBUG\_ENTER(); DEBUG\_STATE(); /\* the main iteration loop for one incoming event \*/ while (!ABSORBED) { /\* iterate event until absorbed \*/ \_class\_particle \_particle\_save; /\* send particle event to component instance, one after the other \*/char flag\_nocoordschange=0; if (!ABSORBED && \_particle->\_index == 1) { /\* begin component arm=Arm() [1] \*/ if (!flag\_nocoordschange) { // flag activated by JUMP to pass coords change if (\_arm\_var.\_rotation\_is\_identity) { coords\_get(coords\_add(coords\_set(x,y,z), \_arm\_var.\_position\_relative),&x, &y, &z); } else mccoordschange(\_arm\_var.\_position\_relative, \_arm\_var.\_rotation\_relative, \_particle); } else flag\_nocoordschange=0; Comp 1 \_particle\_save = \*\_particle; DEBUG\_COMP(\_arm\_var.\_name); DEBUG\_STATE(); class\_Arm\_trace(&\_arm\_var, \_particle); /\* contains EXTEND code \*/ if (\_particle->\_restore) \*\_particle = \_particle\_save; \_particle->\_index++; if (!ABSORBED) DEBUG\_STATE(); } /\* end component arm [1] \*/ if (!ABSORBED && \_particle->\_index == 2) { /\* begin component source=Source\_simple() [2] \*/ if (!flag\_nocoordschange) { // flag activated by JUMP to pass coords change if (\_source\_var.\_rotation\_is\_identity) { coords\_get(coords\_add(coords\_set(x,y,z), \_source\_var.\_position\_relative),&x, &y, &z); } else mccoordschange(\_source\_var.\_position\_relative, \_source\_var.\_rotation\_relative, \_particle); } else flag\_nocoordschange=0; Comp 2 \_particle\_save = \*\_particle; DEBUG\_COMP(\_source\_var.\_name); DEBUG\_STATE(); class\_Source\_simple\_trace(&\_source\_var, \_particle); if (\_particle->\_restore) \*\_particle = \_particle\_save; \_particle->\_index++; if (!ABSORBED) DEBUG\_STATE(); } /\* end component source [2] \*/



#### Main trace Loop (v2) potentially "funnelled"

void raytrace\_all\_funnel(unsigned long long ncount, unsigned long seed) {

```
// set up outer (CPU) loop / particle batches
 unsigned long long loops;
                                                                                                    Smaller "innerloop" due to
 long innerloop=1024*1024; // <-- tune by memory capacity here (max threads: 2147483647);
  loops = ceil((double)ncount/innerloop);
                                                                                                    memory needs for the
 if (ncount>innerloop) {
                                                                                                    array of "particles" etc.
   printf("Defining %1lu CPU loops around kernel and adjusting ncount\n", loops);
   mcset_ncount(loops*2147483647);
} else {
   loops=1:
   innerloop = ncount;
 // outer loop / particle batches
  for (unsigned long long cloop=0; cloop<loops; cloop++) {</pre>
   if (loops>1) fprintf(stdout, "%d..", (int)cloop); fflush(stdout);
   // create particles memory block and pointer array (buffer and sorted)
   #ifdef USE_PGI
   _class_particle* particles = acc_malloc(innerloop*sizeof(_class_particle));
   #else
                                                                                              Allocation of particle array
   _class_particle* particles = malloc(innerloop*sizeof(_class_particle));
   #endif
                                                                                              "fill" the array and make it
   #pragma acc enter data create(particles[0:innerloop])
   // TOD0: _class_particle** psorted = malloc(innerloop*sizeof(_class_particle*));
                                                                                              available to GPU
   // TOD0: _class_particle** pbuffer = malloc(innerloop*sizeof(_class_particle*));
   // TODO: do we need a GPU data section for the above buffers here?
   // set up, generate particles
                                                                                               Initialise / generate particles
   #pragma acc parallel loop present(particles)
                                                                                              GPU side
   for (unsigned long pidx=0 ; pidx < innerloop ; pidx++) {</pre>
     // generate particle state, set loop index and seed
     particles[pidx] = mcgenstate();
     _class_particle* _particle = particles + pidx;
     _particle->_uid = pidx;
     srandom(_hash(pidx + seed)); // _particle->state usage built into srandom macro
   // iterate components
```

//TODO: innerloop = sort\_absorb\_last(particles, innerloop);

```
// arm
                       #pragma acc parallel loop present(particles)
                       for (unsigned long pidx=0 ; pidx < innerloop ; pidx++) {</pre>
         EUROPEAN
         SPALLATION
                         _class_particle* _particle = particles + pidx;
         SOURCE
                         _class_particle _particle_save;
                         if (!ABSORBED) {
                           if (_arm_var._rotation_is_identity)
                             coords_get(coords_add(coords_set(x,y,z), _arm_var._position_relative),&x, &y, &z);
                                                                                                                          Comp 1 in
Main trace
                           else
                                                                                                                          independent kernel
                             mccoordschange(_arm_var._position_relative, _arm_var._rotation_relative, _particle);
Loop (v2)
                           _particle_save = *_particle;
                           class_Arm_trace(&_arm_var, _particle); /* contains EXTEND code */
potentially
                           if (_particle->_restore)
                             *_particle = _particle_save;
"funnelled"
                           _particle->_index++;
                                                                                                        One would then get rid of ABSORB'ed
                       //TODO: innerloop = sort_absorb_last(particles, innerloop);
                                                                                                        particles, do SPLITS here
                       // source
                       #pragma acc parallel loop present(particles)
                       for (unsigned long pidx=0 ; pidx < innerloop ; pidx++) {</pre>
                         _class_particle* _particle = particles + pidx;
                         _class_particle _particle_save;
                         if (!ABSORBED) {
                           if (_source_var._rotation_is_identity)
                             coords_get(coords_add(coords_set(x,y,z), _source_var._position_relative),&x, &y, &z);
                           else
                             mccoordschange(_source_var._position_relative, _source_var._rotation_relative, _particle);
                                                                                                                          Comp 2 in
                           _particle_save = *_particle;
                                                                                                                          independent kernel
                           class_Source_simple_trace(&_source_var, _particle);
                           if (_particle->_restore)
                             *_particle = _particle_save;
                           _particle->_index++;
```



#### Both trace approaches execute the normal component trace function,

#pragma acc routine seq

.... etc

example:

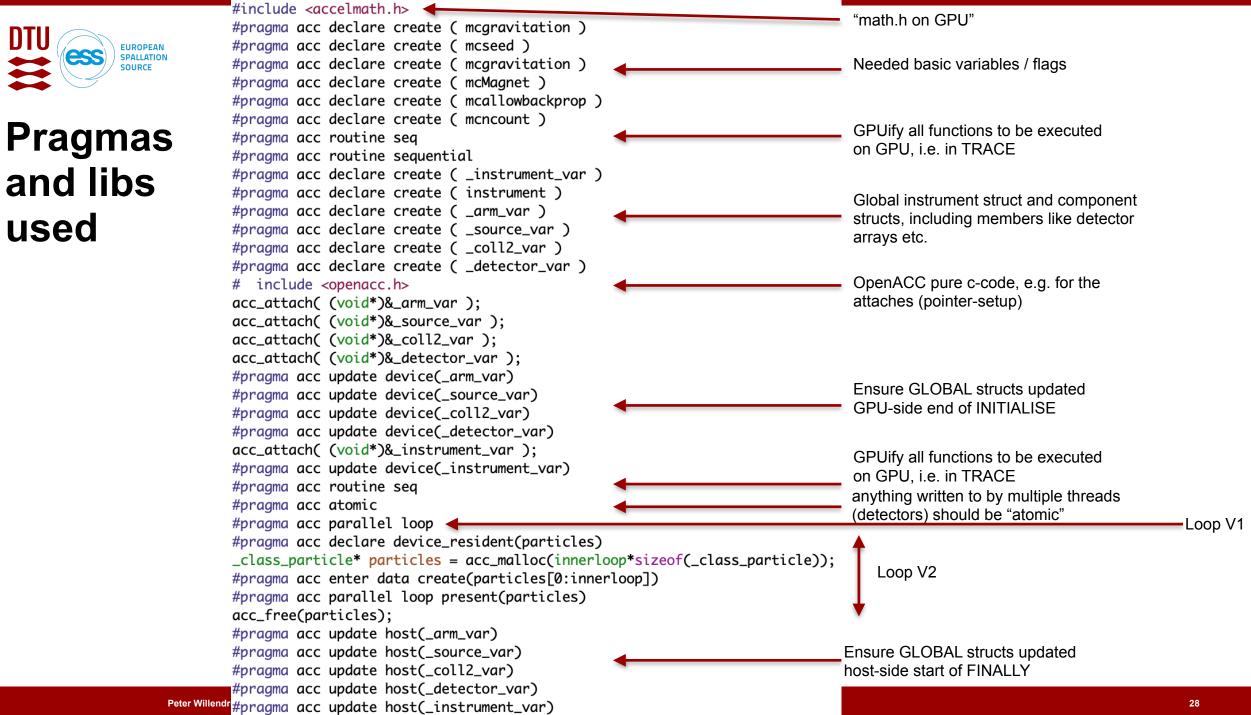
```
_class_Source_simple *class_Source_simple_trace(_class_Source_simple *_comp
  , _class_particle *_particle) {
 ABSORBED=SCATTERED=RESTORE=0;
 #define radius (_comp->_parameters.radius)
 #define yheight (_comp->_parameters.yheight)
 #define xwidth (_comp->_parameters.xwidth)
 #define dist (_comp->_parameters.dist)
 #define focus_xw (_comp->_parameters.focus_xw)
 #define focus_yh (_comp->_parameters.focus_yh)
 #define E0 (_comp->_parameters.E0)
 #define dE (_comp->_parameters.dE)
 #define lambda0 (_comp->_parameters.lambda0)
 #define dlambda (_comp->_parameters.dlambda)
 #define flux (_comp->_parameters.flux)
 #define gauss (_comp->_parameters.gauss)
 #define target_index (_comp->_parameters.target_index)
 #define pmul (_comp->_parameters.pmul)
 #define square (_comp->_parameters.square)
 #define srcArea (_comp->_parameters.srcArea)
 #define tx (_comp->_parameters.tx)
 #define ty (_comp->_parameters.ty)
 #define tz (_comp->_parameters.tz)
 SIG_MESSAGE("[_source_trace] component source=Source_simple() TRACE [Source_simple.comp:127]");
double chi, E, lambda, v, r, xf, yf, rf, dx, dy, pdir;
                                                                                    Contains component trace
t=0;
z=0;
                                                                                    section
if (square == 1) {
  x = xwidth * (rand01() - 0.5);
  y = yheight * (rand01() - 0.5);
} else {
  chi=2*PI*rand01();
                                              /* Choose point on source */
  r=sqrt(rand01())*radius;
                                              /* with uniform distribution. */
  x=r*cos(chi);
  y=r*sin(chi);
randvec_target_rect_real(&xf, &yf, &rf, &pdir,
                         tx, ty, tz, focus_xw, focus_yh, ROT_A_CURRENT_COMP, x, y, z, 2);
```

26



#### **Pro's and cons**

- V1 in general parallelises well, but SPLITS may not be easy to do
- V2 is slower due to multiple kernels and related memory transfer overhead (?). Allows SPLIT
- We are thinking if we should do V3 "best of both worlds", i.e. multiple kernels only when SPLIT is introduced, each SPLIT infers a problem-reduction and another kernel
- Underlying component code identical, only changes in the code generation needed



DTU



### New RNG 'KISS'

- We couldn't easily port Mersenne Twister
- Experimenting with curand showed huge overhead for our relative small number of random numbers
- An RNG 'state' carried with each particle bonus: same seed gives same numbers even when comparing between CPU and GPU
- Required patching prototype of ALL functions making use of e.g. rand01()



#### **Compiler settings used**

pgcc -ta=tesla,managed,deepcopy -Minfo=accel -DUSE\_PGI -DNOSIGNALS -DRNG\_ALG=2

Use our new KISS rng

Disable our signal handling, e.g. USR2 for save. Also the case in our MPI implementation.

Main "enable GPU" switch

Give accel debug information

Used to indicate that copies of variables should be 'deep', e.g. for our structs. (... but removing it seems to have little / no influence?)

Use CUDA shared memory for host-device-host allocation. Needed for our 2D-arrays at present, may include penalty, we could get rid.

Generate Tesla code. "compute capability" e.g. tesla:cc70 may be specified to indicate specific card.



# McStas heading for the GPU... November 2019

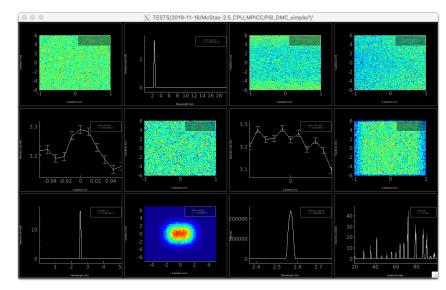
9 instruments fully ported, also realistic ones like PSI\_DMC\* 10-core MPI run, **1e7** in 2 secs

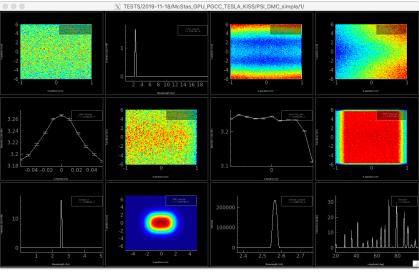


Tesla V100 run, **1e9** in 22 secs

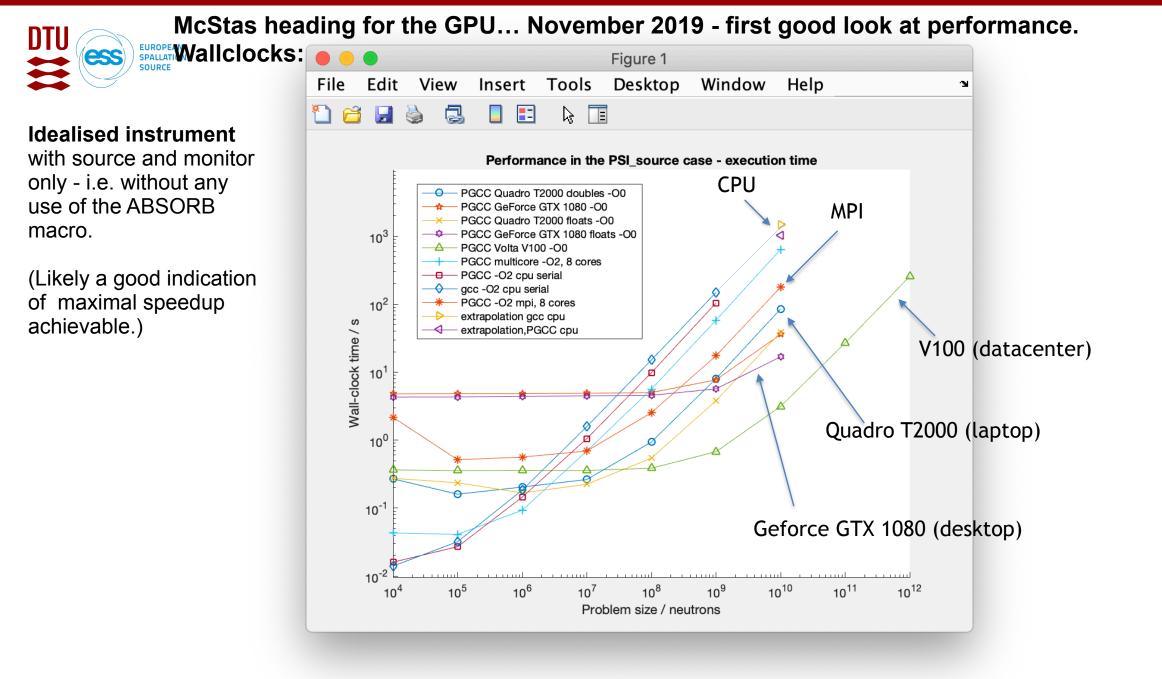
~ i.e. 2 orders of magnitude wrt. a single, modern CPU core

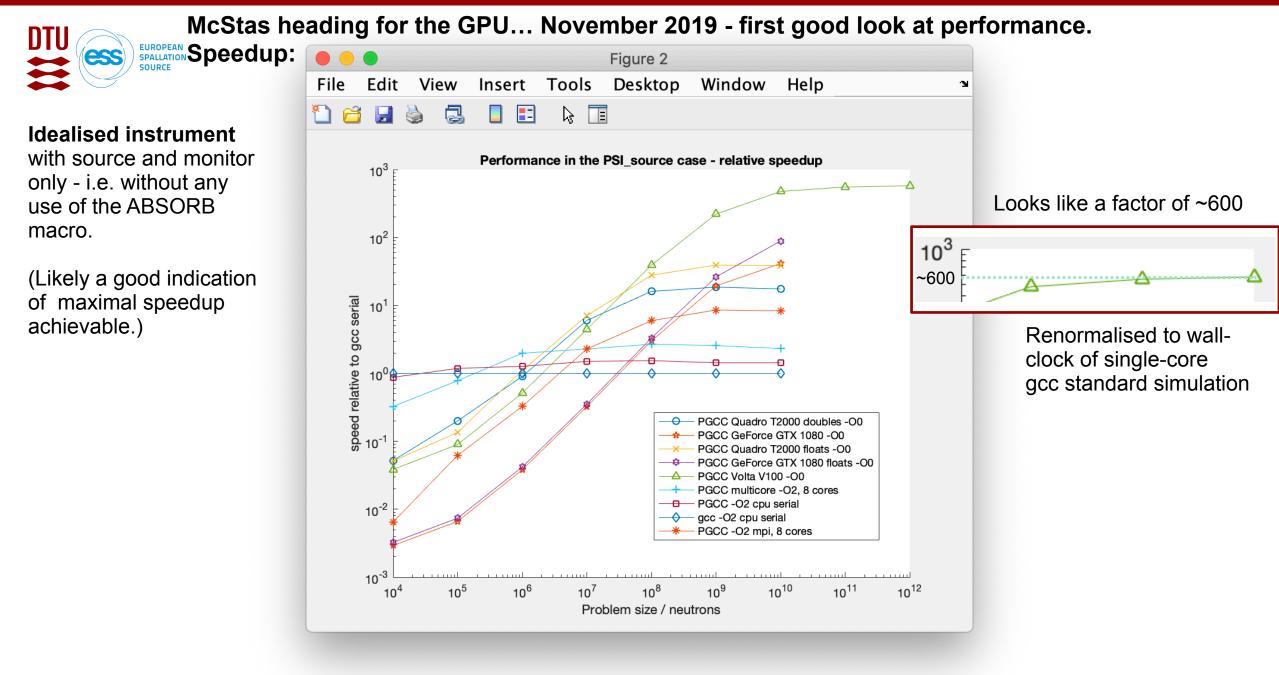






\*Guide component without reflection-file support, SPLIT disabled, OFF geometry disabled







**EUROPEAN** SPALLATION McStas heading for the GPU... December 2019 - today's compilation status:

#### Numerical output with graphics: <u>http://new-nightly.mccode.org/2019-12-06/2019-12-06\_output.html</u>

#### Statistics:

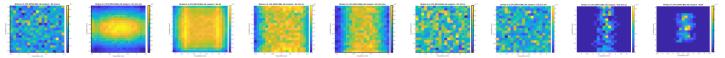
http://new-nightly.mccode.org/2019-12-06/stats.txt

(38 of 142 instruments, 62 of 207 components

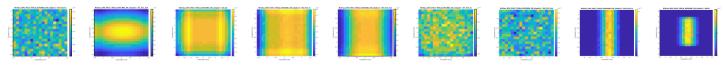
BNL H8 simple/1 - comparison McStas-2.5 Cl	PU_MPICC vs McStas_GPU_PGCC_TESLA_KISS

(Click to access files)

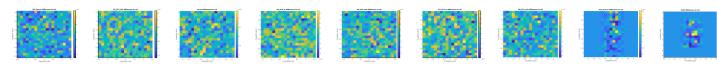
McStas-2.5\_CPU\_MPICC (reference)



#### McStas\_GPU\_PGCC\_TESLA\_KISS



Difference



	McStas- 2.5_CPU_MPICC (ref) - 1e7 n-62-23-6 Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz 20191120_0127_38	McStas_CPU_GCC_KISS - 1e6 n-62-21-99 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz 20191205_0058_36	McStas_CPU_GCC_MT - 1e6 n-62-21-99 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz 20191205_0122_14	McStas_CPU_MPICC_KISS - 1e7 n-62-23-9 Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz 20191205_0034_53	McStas_GPU_PGCC_TESLA_KISS - 1e9 n-62-20-6 Intel(R) Xeon(R) Gold 6126 CPU @ 2.60GHz Tesla V100-PCIE-16GB 20191205_0012_04
BNL_H8_simple	5.80 s   1.99 s   9.7e-10   99%	4.32 s   1.03 s   1.3e-09   136%	4.35 s   0.99 s   1e-09   106%	4.11 s   1.50 s   1.1e-09   110%	17.59 s   8.16 s   9.7e-10   98%



#### McStas 3.0 - next generation code generator - release plans

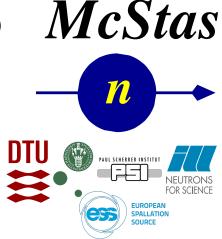
- Limited-functionality "beta" release to be made public soon (jan-mar) after 2.6 (january)
  - Expect bugs!
  - Only a subset of components / instruments
  - Event interchange with 2.6 possible via MCPL

McStas McXtrace

- Main purpose: get this working in 'the wild'
  - · Your instruments will likely require (limited) rewriting
    - Your instruments will need USERVARS for flags that change with each neutron
  - Your own components will likely require rewriting
    - Support for DEFINITION PARAMETERS is deprecated. Use the new string, vector, ... instead

**NVIDIA** 

- New macro MC\_GETPAR2 needed to access other component scopes (NOT via defines)
- E.g. the declare section **cannot include assignments**
- Arrays must be declared/initialized using a new set of functions (i.e. not double PSD\_I[nx][ny] with definition parms)
- Hence some backward compatibility is lost and we need to increment major release #





#### McStas 3.0 - next generation code generator - code camp January 27th-31st in Copenhagen

- Code-camp participants:
  - McStas-McXtrace team (bold means confirmed):
    - Peter Willendrup, Emmanuel Farhi, Erik Knudsen, Jakob Garde, Tobias Weber, Torben R Nielsen, Mads Bertelsen
  - RAMP team from the UK (RAMP is relatively young "McStas equivalent" with GPU-support from the beginning via C++ and OpenCL.)
    - Gino Cassella, Göran Nilsen
  - We didn't attract someone from Nvidia, but we have access to Guido Juckeland in Dresden
- Code-camp focal points:
  - · Port as many remaining instruments / components as possible to GPU
  - Experiment with "telescopic flow"/V2/V3, i.e. a different approach to handling ABSORB's and SPLIT's
  - Experiment with simulation flow between CPU and GPU (a few things can not be done GPU)
  - · Comparisons with RAMP
  - Initiate port of GPU work to McXtrace tree
  - Have fun! :-)

McStas McXtrace

**McStas**