CPS: Overall Code Structure.

A.N. Jackson & S. Booth

Revision: 1.5 Date: 2002/06/06 13:28:25

Contents

1 File types and filename extensions

Type	**QCDSP**				**Elsewhere**	**GNU**
	General	Tartan	T.I.	Sun		
C++ source	.C	-	-	.C	.C	.[cc|cxx|cpp|c++|C]
C/C++ header	.h	-	-	.h	.h	.h
Pre-processed file	.i	-	-	.i	.i	.i
Assembler source	.asm	-	-	.asm	.asm	.[s|S]
Object	-	.tof	.obj	.o	.o	.o
Archive/Library	-	.olb	.lib	.lib	.a	.a
Binary	-	.outtof	.out	.out	.out	-
Output Data	.dat	-	-	.dat	.dat	-
GNU build input files	-	-	-	-	.in	.in

Note that QCDSP also uses linker control files (.lcf), object list files (.ctl), linker map files (.map) and processed assembler files (.lst) [???]. See the QCDSP manual for more information on all the file types used on that platform.

2 ./phys/ - The Physics Code

This is the top level directory. It contains global configuration files (e.g. config.h) and the makefiles.

2.1 ./phys/alg/ - Algorithms

Physical simulation algorithms.

2.2 ./phys/doc/ - Documentation

Documentation for the code, starting at ./phys/doc/index.html.

2.3 ./phys/lib/ - Library Files

All the library archive files from all the other code directories are placed here to be linked against.

2.4 ./phys/mem/ - Memory Routines

Mainly p2v.C, which copies assembler routines into CRAM for the QCDSP machine.

2.5 ./phys/nga/ - Node Gate Array Routines

The hardware interface routines. The global sums and the circular buffer routines are defined in here. This directory also contains the MPI implementation of the SCU layer.

2.6 ./phys/task/ - Task & Scripting Routines

Mainly for QCDSP, these routines parse and execute scripts describing sequences of QCD computations.

2.7 ./phys/tests/ - The Regression Testing Programs

A set of programs for regression testing of the code suite.

2.8 ./phys/util/ - The Physics Utilies Library

This contains the core code, defining the data structures and the essential mathematical algorithms. e.g. the CG solver is defined in here, in context, but called from ./phys/alg/.

3 The GNU build system

When not running on QCDSP, the CPS is built using standard GNU tools to make the platform dependencies as simple to deal with as possible. The tools are autoconf (http://www.gnu.org/manual/autoconf/) and GNU make (gmake, http://www.gnu.org/manual/make/).

3.1 autoconf: resolving platform dependancies and build options

The core of this standard automatic configuration system is the developer's configure script 'configure.in' (in the root ./phys/ directory of the CPS). This defines which files are system/build dependent, and defines macro substitutions to produce versions of those files suitable for a particular build. The configure.in script cannot be run directly, so we need to translate this file into a shell script that will run on as wide a range of platforms as possible. This is what autoconf does; it takes configure.in and produces a shell script called 'configure' which should run on any flavour of UNIX. The user invokes the './configure' command to configure the build.

The build dependant files are:

Input	**Output**	**Description**
Makefile.gnu.in	Makefile.gnu	The makefile for building the CPS library.
Makefile.gnutests.in	Makefile.gnutests	The makefile for building the test suite.
config.h.in	config.h	The global code-configuration header file.
tests/regression.pl.in	tests/regression.pl	A perl script to generate shell scripts for running the test suite.

So, for example, config.h is produced from config.h.in, by performing macrosubstitution of all the @MACRO_NAME@ macros in that file. The full list of macros is:

Macro	**Description**	**Default**
@CC@	The C compiler	"gcc"
@CFLAGS@	Flags for the C compiler	"-O"
@NOT_TESTING_QCDSP@	Is this not QCDSP	"yes"
@PARALLELDEF@	Define the PARALLEL macro	"-DPARALLEL=0"
@MPILINK@	To link with MPI use e.g. -lmpi	""
@PARALLELLINK@	Link with parallel or serial libraries	"SERIALLIB"
@TESTING_PARALLEL@	Should the tests be run as parallel	"no"
@GSUMPRECISION@	Floating-point precision to use for global sums	"double"
@LOCALPRECISION@	Floating-point precision to use elsewhere	"double"
@topwd_srcdir@	Absolute location of the top-level source directory	-

The user can use the configure script to change the behaviour away from the default. Currently, the build options are:

- **–enable-parallel-mpi** which turns the build into a parallel build using the MPI implementation of the SCU.
- **–enable-double-prec="no"** which forces the build to use floats instead of doubles for all calculations except the global summations.

There are other standard ./configure options (use "./configure –help" to see them) but these are not really used in the CPS.

Note that mny of the macros can be overriden using shell environment variables. For example, if you want to use the Sun C++ compiler with debugging information switched on, i.e. CC -g , you should use (in bash):

```
export CC=CC
export CFLAGS=-g
./configure
```

You should be warned, however, that the configure script stores a record of the build options in a file called config.status or config.cache. You may have to remove this file to force the build behaviour to change.

There are a number of other special files that autoconf creates to store information and to ensure that the ./configure system will run anywhere. They are config.cache, config.guess, config.log, config.status, config.sub, install-sh, missing, mkinstalldirs, uname. See the autoconf manual for detailed information on what these files do.

3.2 gmake: the GNU make program

The GNU version of make is used simply because it has a little more functionality that standard make programs. In particular, the makefiles can be (and are) recursive.

4 Outline of the new code structure

4.1 A unified build system

4.2 Cross-platform portability

4.3 Cross-platform testing suite