# The QCDIO API

A.N. Jackson & S. Booth

Revision: 1.4 Date: 2001/10/23 14:45:08

## Contents

# 1 Introduction

This document outlines our current proposal for the QCDIO API, as (to be) implemented in the Columbia code. The code implementing this API can fe found at:

- ./phys/util/include/qcdio.h
- ./phys/util/qcdio/qcdio.C

# 2 Standard Output

By default, when running of QCDSP, only the output from node 0 is returned to Q shell. In order to reproduce this behaviour elsewhere (for regression testing purposes), the qcdio.h file overrides [f]printf to map it onto q[f]printf which are defined in qcdio.C.

# 3 Loading/Saving Gauge Configurations

Again, for testing purposes, we need a simple API for loading and saving the gauge configurations. The suggested form is:

- qload_gauge( char* filename_prefix, Lattice lat ) Load the configuration from a set of files specified by the UKQCD filename prefix into the Lattice object lat.
- qload_gauge( char* filename_prefix, Lattice lat) Save the current gauge configuration from the Lattice object lat to a set of files using the specified UKQCD filename prefix.

EPCC will write a very simple gauge configuration reader for the Columbia code. Because of the DOE SciDAC project, there is no point in putting too much work into modifying the infrastructure of the code. The plan would be to just read the configurations in UKQCD format.

Craig McNeile will write a conversion code to convert the following formats:

- NERSC
- MILC
- SZIN

into ukqcd format (using http://www.ph.ed.ac.uk/ukqcd/public/misc.html as a starting point).

## 3.1 The UKQCD gauge configuration file format

In general a ”configuration” is tar (UNIX tape archive) file with name

```
D<Beta>C<Clover>K<Kappa>U<Trajectory>.tar
```

where

- Beta is the beta value without the decimal point.
- Clover is the clover value without the decimal point (probably rounded)
- Kappa is the kappa_sea value without the ’0.1’ at the start
- Trajectory is a 6 digit number designating the HMC trajectory.

For example D52C202K3500U007000.tar contains the information for trajectory serial no 7000 with

```
beta = 5.2
clover ~ 2.02 (as to how to find the exact value see later)
kappa\_sea = .1350
```

This tar file contains the following files:

`D<Beta>C<Clover>K<Kappa>U<trajectory>T<timeslice>` – these are the gauge timeslices

`D<Beta>C<Clover>K<Kappa>U<trajectory>PT<timeslice>` – these are the conjugate momenta timeslices

`D<Beta>C<Clover>K<Kappa>U<Trajectory>.par` – The parameter file

`D<Beta>C<Clover>K<Kappa>U<Trajectory>.rng` – The random number state

The random number state and the conjugate momenta are probably not useful for analysis, their purpose in the parameter file was just to allow consistent restarts from the configuration using the HMC code should we have wished to do so.

The parameter file is useful as it contains all the simulation parameters such as the precise values of beta, c_sw, and kappa_sea which may have been truncated to keep filenames a manageable length. It also contains the lattice dimensions and validation information for the configuration such as the plaquette.

The parameters plaquette_real, plaquette_image refer to the plaquette over the whole configuration whereas the parameters tplaquette_real[index], tplaquette_imag[index] refer to the spatial plaquette on timeslice ¡index¿

The gauge configurations are saved in timeslices in files with names

`D<Beta>C<Clover>K<Kappa>U<Trajectory>T<Timeslice - always 2 digits>`

The precision of the saved gauges is usually 4 bytes (This is `REAL, KIND=4` in Fortran 90 and `sizeof(float)` in C for most architectures. The ghmc code can save with 8 bytes precision also, but this is almost never used)

The byte ordering of the configurations is the cray t3e byte ordering. [I believe this is "big-endian" - ANJ]. This is the same as the byte ordering for Sun workstations but is the opposite of the byte ordering for the alpha systems.

WIthin a timeslice array indexing runs as follows (fastes index first)

complex components, rows (2 rows only), columns, direction, x, y, z

| | | | |
|---|---|---|---|
| | complex components | = 0 - 1 | (Real Part = 0, Imaginary Part = 1) |
| | rows | = 0 - 1 | (Two rows only) |
| | columns | = 0 - 2 | (Three columns) |
| where | direction(mu) | = 0 - 3 | |
| | x | = 0 - Latt_x-1 | |
| | y | = 0 - Latt_y-1 | |
| | z | = 0 - Latt_y-1 | |

Where Latt_x, Latt_y, Latt_z are sizes of the lattice in the various directions. In our production runs these are always 16.

The last row of the gauges has to be regenerated as it is not stored.

Thus to read in a gauge configuration one must read

4 * 2 * 2 * 3 * 4 * Latt_x * Latt_y * Latt_z

bytes into a buffer

(These are precision * No of complex components * 2 rows * 3 columns * 4 directions * Spatial dimensions in order )

Byte swap the buffer (depending on architecture) if necessary

Pack the buffer away into your lattice data structure in memory usually by looping over all the components in the order described above (complex fastest, z slowest)

### 3.1.1 Regenerate the third row

Include is a set of very simple C++ routines that can be used to handle gauges on a workstation. Our Fortran code is also available but is horribly complicated as it has been designed for MPP use and involves a lot of preprocessing, conditional compilation, header files, dealing with processor layout etc which are much harder to glean information from. This code is available as gauge.tar.gz on the web page.

The most important of all these files is gauge.cpp which is the code for all the methods of the SU3GaugeTimeslice and the program gaugetest.cpp which illustrates how the class can be used to read in a gauge. The rest of the classes are just support ( a very primitive complex number class,

and an su3 matrix class – also primitive ). One should be able to compile gaugetest.cpp using the Makefile supplied. It has been tested with Dec C++ on an Alpha running Digital Unix.

The SU3Matrix class illustrates the technique used to regenerate the third row. The swap.cpp subroutines carry out the byte swapping.

To change the C++ compiler to any other, edit the Makefile and change the CC macro to the C++ compiler of your choice. Currently it is set to cxx.

For suns the byte swap flag in gaugetest.cpp should be set to 0.

To compile the test program copy the test file gauge.tar.gz to its final location.

Unzip it using gunzip:

```
$ gunzip gauge.tar.gz
```

Untar it

```
$ tar xvf gauge.tar
```

It should create its own directory called gauge and untar there. To compile go into this directory and make:

```
$ cd gauge; make
```

The code can be run to read in a gauge configuration by typing

```
$ ./gaugetest <Xsize> <Ysize> <Zsize> <Tsize> <ByteSwap> <Prefix>
```

Xsize, Ysize, Zsize are the lattice spatial dimensions in the directions of x, y, and z respectively.

Tsize is the number of timeslices

Byte swap is a flag indicating wether or not the gauge should have its byte order reversed.

Prefix is the path of the gauge timeslice files without the T¡timeslice¿

Eg to read a $16^3$ x 32 configuration in the current directory with filename stem of D52C202K3500U007000 on an Alpha where one has to reverse the byte order of the floating point numbers ons could run

```
$ ./gaugetest 16 16 16 32 1 ./D52C202K3500U007000
```

If all goes well you should get output of the following nature:

```
Reading timeslice ./D52C202K3500U007000T00
Re Tr Plaq = 0.533327  Im Tr Plaq = 0.000494551
Reading timeslice ./D52C202K3500U007000T01
Re Tr Plaq = 0.528731  Im Tr Plaq = -0.000777869
```

```
Reading timeslice ./D52C202K3500U007000T02
Re Tr Plaq = 0.533354  Im Tr Plaq = 0.000177485
Reading timeslice ./D52C202K3500U007000T03
Re Tr Plaq = 0.530886  Im Tr Plaq = -0.000693423
Reading timeslice ./D52C202K3500U007000T04
.
. (etc)
.
```

If gaugetest dies with a segfault or gets the plaquette wrong it can be because the gauge dimensions are wrong or that the byte swap flag is set incorrectly or that something horrible happened on the way to the transport of the gauge or that my C++ is not as portable as I like to think

If you have problems even despite this assistance pleas contact me via email at B.Joo@ed.ac.uk