Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

(Re-)Configurable Data File Writing At UKIRT

M. H. Purves

Joint Astronomy Centre, 660 N. A'ohoku Pl., Hilo, HI 96720

Abstract. Recent development at UKIRT has shown that re-configurable data file writing isn't only quite possible but also advantageous in the short as well as in the longer run.

1. Introduction

The writing of astronomical data files developed over time. At first we had single element detectors, and computing power and disk space was expensive, and tape storage was bulky. We were writing data files with little extra computing overhead, and saving space in storing the data; getting the data stored was what counted, not how it was written. As the number of detector elements increased, so did computing power. Memory and disk space got cheaper, tapes got smaller. All the considerations that drove data file writing in the beginning became less important, but something else became more important at the same time: maintainability and portability, as well as, at least in our case, the wishes of the scientists to change things around as instrument configurations change.

2. History

Before CGS3 (commissioned 1990), data file writing was integrated in the data acquisition software and basically as little data as possible was written. Of course there was very little data to write at the time. As far a I know, CGS3 was the first UKIRT instrument to have a dedicated data recording task.

IRCAM and CGS4 have been sharing software since 1995. The CGS4/ IRCAM system has a dedicated data file writing task and data file writing is queued.

Part of the underlying code looks like this:

```
CALL ALF_FITS_PUTOC(1,'TELESCOP',TELESCOPE,'Telescope name',STATUS)
CALL ALF_FITS_PUTOC(2,'INSTRUME',INSTRUMENT,'Instrument',STATUS)
CALL ALF_FITS_PUTOC(3,'OBSERVER',OBSERVERS,'Observer name(s)',STATUS)
CALL ALF_FITS_PUTOC(4,'OBSREF',REFERENCE,'Observer reference',STATUS)
CALL ALF_FITS_PUTOC(5,'DETECTOR',DETECTOR,'Detector array used',STATUS)
CALL ALF_FITS_PUTOC(6,'OBJECT',OBJECT,'Name of observed object',STATUS)
CALL ALF_FITS_PUTOC(7,'OBSTYPE',OBS_TYPE,'Type of observation',STATUS)
```

This means that the order of FITS header items is the same between the two instruments. Modifications, as well as accommodating differences between the two instruments is done in the code and is difficult. We have had to deal with new keywords as well as obsolete keywords over the years, so when we moved to a completely different system for the UKIRT Fast-Track Imager, UFTI (commissioned in 1998) we took some time to rethink this.

3. Present

The UFTI data acquisition system was the first at UKIRT to go into operation under VxWorks/EPICS running in a VME crate. UFTI's data files are currently written directly from its control/acquisition VME crate to a NFS-mounted disk on a Sun. They are currently FITS files, using CFITSIO on VxWorks. This works well after overcoming some glitches in the beginning. The sequence of header keywords is (unfortunately) not table-driven, but resides in a C header include file. There is one structure for each keyword, and each keyword structure is complete in itself. The sequence of keyword structures in the header structure determines the sequence in which the keywords are written.

```
{"INSTRUME", TSTRING, NULL, 1, 0, 0, "Instrument"         ,{""}},
{"OBSERVER", TSTRING, NULL, 1, 0, 0, "Observer name(s)"   ,{""}},
{"OBSREF",   TSTRING, NULL, 1, 0, 0, "Observer reference" ,{""}},
{"OBSNUM",   TINT,    NULL, 1, 0, 0, "Observation number" ,{""}}
```

While the file is easy to change, this part of the software still needs to be recompiled for changes to take effect.

Not every keyword in the structure is necessarily written to the data file. The content of the header is determined by the contents of the EPICS database.

The underlying code is simple:

```
for ( i = 0; i < NUM_FITS_KEYS; i++ )
{
   if (dhsWriteKeys[i].fitsItem &&
       dhsWriteKeys[i].used        )  /* else do nothing here */ {
       if (!strncmp (dhsWriteKeys[i].keyName, "BLANKLIN", 8))
       {
          if (fits_write_record(fptr,blankkey, &status))
                   dhsErrHd (status);
       }
       else if (dhsWriteKeys[i].keyType == TDOUBLE)
       {
          if (fits_write_key_dbl(fptr, dhsWriteKeys[i].keyName,
                                 dhsWriteKeys[i].uval.dval,
                                 dhsWriteKeys[i].keyPrec,
                                 dhsWriteKeys[i].keyComment,
                                 &status)) dhsErrHd (status);
       }
       else if (!strncmp (dhsWriteKeys[i].keyName, "COMMENT",7))
       {
          if (fits_write_comment(fptr, dhsWriteKeys[i].keyComment,
                                 &status)) dhsErrHd (status);
```

```
        }
        else
        {
           if (fits_write_key(fptr, dhsWriteKeys[i].keyType,
                              dhsWriteKeys[i].keyName,
                              (void*) &dhsWriteKeys[i].uval,
                              dhsWriteKeys[i].keyComment,
                              &status )) dhsErrHd (status);
        }
 } }
```

This same code writes the data files for UFTI as well as for UKIRT's Wavefront Sensor. The Wavefront Sensor has a much shorter FITS header, easily accomplished with a change to the EPICS database.

4. Future - Where do we Go from Here?

With the advent of ORAC (Bridger et al. 2000) we have started using the Gemini DHS (Hill & Gaudet 1998) modified by Alan Pickup for UKIRT to write HDS container files. Since the Wavefront Sensor will use the same software in its VME crate and still write FITS files we have modified the software originally written for UFTI so that it can either hand data to the DHS or write FITS files:

```
for ( i = 0; i < NUM_FITS_KEYS ; i++ ) {
   if (dhsWriteKeys[i].fitsItem &&
       dhsWriteKeys[i].used        )  {
      if (!strncmp (dhsWriteKeys[i].keyName, "BLANKLIN", 8)) {
         if ( writingFits ) {
            if (fits_write_record(fptr,blankkey,&status))
                     dhsErrHd(status);
         } else       {         /* we do nothing */         }
      }
      else if (dhsWriteKeys[i].keyType == TDOUBLE)   {
         if ( writingFits ) {
         if (fits_write_key_dbl(fptr, dhsWriteKeys[i].keyName,
                                dhsWriteKeys[i].uval.dval,
                                dhsWriteKeys[i].keyPrec,
                                dhsWriteKeys[i].keyComment,
                                &status)) dhsErrHd (status);
         } else {
         dhsBdAttribAdd (dataset, dhsWriteKeys[i].keyName,
                         DHS_DT_DOUBLE, 0, NULL,
                         dhsWriteKeys[i].uval.dval, &mydhsStatus);
         if (mydhsStatus!=DHS_S_SUCCESS)
                         dhsDhsErrHd (mydhsStatus);
         }
      }
      else if (!strncmp (dhsWriteKeys[i].keyName, "COMMENT", 7)) {
         if ( writingFits ) {
```

```
              if(fits_write_comment(fptr,dhsWriteKeys[i].keyComment,
                                     &status)) dhsErrHd (status);
          } else          {       /* do nothing here */ }
      } else {
         if ( writingFits ) {
            if (fits_write_key(fptr, dhsWriteKeys[i].keyType,
                               dhsWriteKeys[i].keyName,
                               (void*) &dhsWriteKeys[i].uval,
                               dhsWriteKeys[i].keyComment,
                               &status )) dhsErrHd (status);
         } else {
            if (dhsWriteKeys[i].keyType == TINT) {
               dhsBdAttribAdd (dataset, dhsWriteKeys[i].keyName,
                               DHS_DT_INT32, 0, NULL,
                               dhsWriteKeys[i].uval.lval,
                               &mydhsStatus);
               if(mydhsStatus!=DHS_S_SUCCESS)
                               dhsDhsErrHd(mydhsStatus);
            } else {
               dhsBdAttribAdd (dataset, dhsWriteKeys[i].keyName,
                               DHS_DT_STRING, 0, NULL,
                               dhsWriteKeys[i].uval.cval,
                               &mydhsStatus);
               if(mydhsStatus!=DHS_S_SUCCESS)
                               dhsDhsErrHd(mydhsStatus);
            }
         }
      }
  }
}

(some line feeds changed for space reasons)
```

This still retains the same configurability for which we originally aimed. The DHS provides queued data handling and file writing, eliminating the overheads that the current fast track configurable approach presents.

Acknowledgments. Among others, UKIRT instrument code was written by: Colin Aspin, Jeremy Bailey, Steven Beard, Alan Bridger, Phil Daly, Clive Davenhall, Kevin Krisciunas, Len Lawrence, John Lightfoot, Bernard McNally, Magnus Paterson, Alan Pickup, Nick Rees, Ian Sheffield, Malcolm Stewart

References

Bridger, A., et al. 2000, this volume, 467

Hill, N., Gaudet, S. 1998, "ICD3 - Bulk Data Transfer", Gemini Data Handling System Report dhs-pdr-icd3/23