GENPRO-1 File Format Description

Nicholas DeCicco

July 5, 2016

0.1 Introduction

The GENPRO-1 file format is a binary file format for storing arbitrary observational data at intervals.

0.1.1 About GENPRO-I Observational Data from NCAR EOL

Files with a "-C" suffix are COS-blocked files. Files without a "-C" suffix are Ampex TMS-4 Terabit Memory System (TBM) files.

0.2 Header

All GENPRO-1 files begin with a plain-text header text header using a 6-bit word size and custom encoding. The header contains no newline characters. The header is divided into two parts: the first part is 1100 (6-bit) characters long and contains:

- The number of parameters contained in the file.
- The cycle period.
- The word size used for data.

Note that there does not appear to be an ideal line width to display this portion of the header. All of these values in this part of the header are at fixed offsets from the start of the file. Some of these offsets are listed in tbl. 1.

Table 1: Zero-based header offsets.

Attribute	Start	End	Length	C format string
File description	0	31	32	N/A
Date	24	30	7	`"%d%3s%2d"`
Number of parameters	175			`"%3d"`
Total number of samples per cycle	246			`"%4d"`
Cycle period	290			`"%f"`

The second part of the header consists of one 100 character line (again, with no newline characters) for each parameter ("variable" in NetCDF parlance) in the file. Each of these lines contains, in order:

1. The one-based index of the parameter.
2. The sample rate of the parameter, in multiples of the sampling frequency. E.g., a parameter with a sample rate of 20 samples/cycle belonging to a dataset with a cycle period of 0.5 s would mean an actual sample rate for that parameter of $20/(0.5\,\mathrm{s}) = 40\,\mathrm{Hz}$.
3. A human readable description of the parameter.
4. A short name for the parameter.
5. The units for the parameter.
6. The scale and offset ("bias") for each parameter (field).

Here is an example header:

```
492B-03  PHOENIX - 78   05SEP78 14/37/00          FIRST TIME ON THIS FILE 14 37  0       THIS FILE I
S ALL OR PART OF TIME PERIOD 14 37  0 TO 16 23 45  DESCRIPTION OF RECORD --  67 PARAMETERS WERE SAVE
D AT THEIR RESPECTIVE RATES.  THIS REPRESENTS 1226 SAMPLES/PROGRAM CYCLE WHERE A CYCLE IS 1.000 SEC
 THE  1 CYCLES OF1226 SAMP/CYC = 1226 WERE THEN SCALED INTO 20 BIT INTEGERS AND PACKED 3 SAMPLES/WOR
D INTO 409 60 BIT PACKED WORDS          --METHOD OF SCALING--  A BIAS AD(I) WAS ADDED TO EACH SAMPLE
 OF EACH PARAMETER  TO ELIMINATE ANY NEGATIVE VALUES. THE BIASED SAMPLE WAS THEN MULTIPLIED BY P(I)
TO INSURE THE PROPER NUMBER OF DECIMAL PLACES WERE SAVED.  THE PACKED RECORD MAY BE UNPACKED BY RIGH
T JUSTIFYING 20 BITS AT A TIME AND REVERSING THE ABOVE SCALING PROCESS. AS EXAMPLE, S(I)=N/P(I)-AD(I
), WHERE N IS THE 20 BIT SCALED INTEGER, S(I) THE DESIRED UNSCALED PARAMETER, AD(I),P(I) THE CORRESP
ONDING SCALE FACTORS  THE ORDER, RATE, PLOT TITLE, PRINT LAB, UNITS, AD AND P SCALE FACTORS OF EACH
```

```
PARAMETER FOLLOW
  1)    1     PROCESSOR TIME (SECONDS) AFTER MIDNIGHT    TIME      SEC     = (N/    1.0) -    0.0
  2)    1     UNALTERED TAPE TIME (SEC) AFTER MIDNIGHT   TPTIME    SEC     = (N/    1.0) -    0.0
  3)    1     LTN-51 ARINC TIME LAG (SEC)                TMLAG     SEC     = (N/ 1000.0) -  100.0
  4)    1     EVENT MARKER WORD                          EVMRKS              = (N/    1.0) -    0.0
  5)    1     PILOT MICROPHONE SWITCH (XMIT) (VDC)       XMIT      VDC     = (N/ 1000.0) -  100.0
  6)    1     FIXED ZERO VOLTAGE (VDC)                   FZV       VDC     = (N/ 1000.0) -  100.0
  7)   20     RAW INS LATITUDE (DEG)                     ALAT      DEG     = (N/ 1000.0) -  100.0
  8)   20     RAW INS LONGITUDE (DEG)                    ALONG     DEG     = (N/ 1000.0) -  200.0
  9)   20     AIRCRAFT TRUE HEADING (ARINC) (DEG)        THI       DEG     = (N/ 1000.0) -  100.0
 10)   20     INS WANDER ANGLE (DEG)                     ALPHA     DEG     = (N/ 1000.0) -  100.0
 11)   20     RAW INS GROUND SPD X COMPONENT (M/S)       XVI       M/S     = (N/ 1000.0) -  500.0
 12)   20     RAW INS GROUND SPD Y COMPONENT (M/S)       YVI       M/S     = (N/ 1000.0) -  500.0
 13)   20     RAW INS GROUND SPEED (M/S)                 GSF       M/S     = (N/ 1000.0) -  100.0
 14)   20     AIRCRAFT PITCH ATTITUDE ANGLE (DEG)        PITCH     DEG     = (N/ 1000.0) -  100.0
 15)   20     AIRCRAFT ROLL ATTITUDE ANGLE (DEG)         ROLL      DEG     = (N/ 1000.0) -  100.0
 16)   20     AIRCRAFT TRUE HEADING (YAW) (DEG)          THF       DEG     = (N/ 1000.0) -  100.0
 17)   20     RAW INS VERTICAL VELOCITY (M/S)            VZI       M/S     = (N/ 1000.0) -  500.0
 18)   20     RAW DYNAMIC PRESSURE (WING) (MB)           QCW       MB      = (N/ 1000.0) -  100.0
 19)   20     RAW DYNAMIC PRESSURE (GUST PROBE) (MB)     QCG       MB      = (N/ 1000.0) -  100.0
 20)   20     CORRECTED DYNAMIC PRESR (WING)  (MB)       QCWC      MB      = (N/ 1000.0) -  100.0
 21)   20     CORRECTED DYNAMIC PRESR (GUST PROBE)(MB)   QCGC      MB      = (N/ 1000.0) -  100.0
 22)   20     RAW STATIC PRESSURE (FUSELAGE) (MB)        PSF       MB      = (N/ 1000.0) -    0.0
 23)   20                 *** UNUSED ***                 UNUSED              = (N/    1.0) -    0.0
 24)   20     CORRECTED STATIC PRESR (FUSELAGE) (MB)     PSFC      MB      = (N/ 1000.0) -    0.0
 25)   20                 *** UNUSED ***                 UNUSED              = (N/    1.0) -    0.0
 26)   20     NACA PRESSURE ALTITUDE (M)                 HP        M       = (N/   10.0) -  500.0
 27)   20     GEOMETRIC (RADIO) ALTITUDE (M)             HGM       M       = (N/ 1000.0) -  100.0
 28)   20     TOTAL TEMPERATURE (WING ROSEMOUNT) (C)     TTW       C       = (N/ 1000.0) -  100.0
 29)   20     TOTAL TEMPERATURE (BOOM ROSEMOUNT) (C)     TTB       C       = (N/ 1000.0) -  100.0
 30)   20     TOTAL TEMPERATURE (FAST RESPONSE) (C)      TTKP      C       = (N/ 1000.0) -  100.0
 31)   20     AMBIENT TEMP (WING ROSEMOUNT) (C)          ATW       C       = (N/ 1000.0) -  100.0
 32)   20     AMBIENT TEMP (BOOM ROSEMOUNT) (C)          ATB       C       = (N/ 1000.0) -  100.0
 33)   20     AMBIENT TEMPERATURE (FAST RESPONSE) (C)    ATKP      C       = (N/ 1000.0) -  100.0
 34)   20                 *** UNUSED ***                 UNUSED              = (N/    1.0) -    0.0
 35)   20     DEW/FROSTPOINT TEMP (THERMOELEC) (C)       DP        C       = (N/ 1000.0) -  100.0
 36)   20     DEWPOINT TEMPERATURE (THERMOELEC) (C)      DPC       C       = (N/ 1000.0) -  100.0
 37)   20     ABSOLUTE HUMIDITY (THERMOELEC) (G/M3)      RHOTD     G/M3    = (N/ 1000.0) -  100.0
 38)   20     REFRACTIVE INDEX (N-UNITS)                 RFI       N       = (N/ 1000.0) -  100.0
 39)   20     ABSOLUTE HUMIDITY (REFRACT) (G/M3)         RHORF     G/M3    = (N/ 1000.0) -  100.0
 40)   20     AIRCRAFT TRUE AIRSPEED (WING) (M/S)        TASW      M/S     = (N/ 1000.0) -  100.0
 41)   20     AIRCRAFT TRUE AIRSPEED (GUST) (M/S)        TASG      M/S     = (N/ 1000.0) -  100.0
 42)   20     ATTACK ANGLE (FIXED VANE) (DEG)            AFIX      DEG     = (N/ 1000.0) -  100.0
 43)   20     ATTACK ANGLE (ROTATING VANE) (DEG)         AROT      DEG     = (N/ 1000.0) -  100.0
 44)   20     SIDESLIP ANGLE (FIXED VANE) (DEG)          BFIX      DEG     = (N/ 1000.0) -  100.0
 45)   20     SIDESLIP ANGLE (ROTATING VANE) (DEG)       BROT      DEG     = (N/ 1000.0) -  100.0
 46)   20     GUST PROBE TIP VERT ACCEL (M/S2)           VAC       M/S2    = (N/ 1000.0) -  100.0
 47)   20     GUST PROBE TIP LATERAL ACCEL (M/S2)        LAC       M/S2    = (N/ 1000.0) -  100.0
 48)   20     WIND VECTOR EAST GUST COMPONENT (M/S)      UI        M/S     = (N/ 1000.0) -  200.0
 49)   20     WIND VECTOR NORTH GUST COMPONENT (M/S)     VI        M/S     = (N/ 1000.0) -  200.0
 50)   20     WIND VECTOR VERTICAL GUST COMP (H) (M/S)   WI        M/S     = (N/ 1000.0) -  100.0
 51)   20     WIND VECTOR LNGTDNL GUST COMPONENT (M/S)   UX        M/S     = (N/ 1000.0) -  200.0
 52)   20     WIND VECTOR LATERAL GUST COMPONENT (M/S)   VY        M/S     = (N/ 1000.0) -  200.0
 53)   20     HORIZONTAL WIND DIRECTION (DEG)            WDRCTN    DEG     = (N/ 1000.0) -  100.0
 54)   20     HORIZONTAL WIND SPEED (M/S)                WSPD      M/S     = (N/ 1000.0) -  100.0
 55)   20     RAW INS GROUND SPD EAST COMP (M/S)         VEW       M/S     = (N/ 1000.0) -  500.0
 56)   20     RAW INS GROUND SPD NORTH COMP (M/S)        VNS       M/S     = (N/ 1000.0) -  500.0
 57)   20     DISTANCE EAST OF BAO TOWER (KM)            DEIBAO    KM      = (N/  100.0) - 1000.0
 58)   20     DISTANCE NORTH OF BAO TOWER (KM)           DNIBAO    KM      = (N/  100.0) - 1000.0
 59)   20     AIRCRAFT C.G. ACCELERATION (M/S2)          CGAC      M/S2    = (N/ 1000.0) -  100.0
 60)   20     DAMPED AIRCRAFT VERT VELOCITY (M/S)        WP3       M/S     = (N/ 1000.0) -  100.0
 61)   20     PRESSURE-DAMPED INERTIAL ALTITUDE (M)      HI3       M       = (N/   10.0) -  500.0
 62)   20     AIRCRAFT INDICATED AIRSPEED (GUST) (M/S)   IASG      M/S     = (N/ 1000.0) -  100.0
 63)   20       MIXING RATIO (G/KG)                      RM        G/KG    = (N/ 1000.0) -  100.0
 64)   20     SPECIFIC HUMIDITY (G/KG)                   SPHUM     G/KG    = (N/ 1000.0) -  100.0
 65)   20      POTENTIAL TEMPERATURE (K)                 THETA     K       = (N/ 1000.0) -  100.0
 66)   20     VIRTUAL POTENTIAL TEMPERATURE (K)          VTHETA    K       = (N/ 1000.0) -  100.0
 67)   20     DEWPOINT TEMPERATURE (REFRACT) (C)         DPCRF     C       = (N/ 1000.0) -  100.0
```

Offsets from the start of a line to each attribute associated with a parameter are listed in tbl. 2

Table 2: Parameter attribute offsets and lengths. Offsets are zero-based. End offsets are inclusive.

Attribute	Start	End	Length
Index	0	2	3
Sample rate	4	7	4
Description	13	54	42
Short name	56	64	9
Units	66	72	7
Scale factor	80	85	6
Bias	90	95	6

0.2.1 6-bit Character Encoding

The mapping between the 6-bit character encoding and ASCII character codes is shown in tbl. 3. As an example, the first 12 bytes (96 bits or 16 6-bit words) of a file in the PHOENIX-78 dataset (similar to the above example) are `7e47 4299 b72d b502 0f14 e258`. Regrouping this byte sequence into groups of three nibbles (12 bits), we decode these new groupings of three into bits, regroup the bits into groups of 6, then convert the groupings of 6 into their decimal equivalents, and look up the ASCII values corresponding to these decimal values in tbl. 3:

Hex	Binary				6-bit word values		ASCII	
`7e4`	`0111`	`11`	`10`	`0100`	31	36	`4`	`9`
`742`	`0111`	`01`	`00`	`0010`	29	2	`2`	`B`
`99b`	`1001`	`10`	`01`	`1011`	38	27	`-`	`0`
`72d`	`0111`	`00`	`10`	`1101`	28	45	`1`	`␣`
`b50`	`1011`	`01`	`01`	`0000`	45	16	`␣`	`P`
`20f`	`0010`	`00`	`00`	`1111`	8	15	`H`	`O`
`14e`	`0001`	`01`	`00`	`1110`	5	14	`E`	`N`
`258`	`0010`	`01`	`01`	`1000`	9	24	`I`	`X`

So the 12 byte sequence `7e47 4299 b72d b502 0f14 e258` translates into the 16 character sequence `492B-01  PHOENIX` (note the two spaces between "`492B-01`" and "`PHOENIX`").

Table 3: GenPro Character Encoding

GENPRO-1			ASCII			
Dec	Hex	Oct	Hex	Dec	Oct	Character
0	0	0	58	3A	72	:
1–26	1–1A	1–32	66	42	102	A–Z
27–36	1B–24	33–44	48	30	60	0–9
37	25	45	43	2B	53	+
38	26	46	45	2D	55	-
39	27	47	42	2A	52	*
40	28	50	47	2F	57	/
41	29	51	40	28	50	(
42	2A	52	41	29	51	)
43	2B	53	36	24	44	$
44	2C	54	61	3D	75	=
45	2D	55	32	20	40	Space
46	2E	56	44	2C	54	,
47	2F	57	46	2E	56	.
48	30	60	35	23	43	#
49	31	61	91	5B	133	[
50	32	62	93	5D	135	]
51	33	63	37	25	45	%
52	34	64	34	22	42	"
53	35	65	95	5F	137	_
54	36	66	33	21	41	!
55	37	67	38	26	46	&
56	38	70	39	27	47	’
57	39	71	63	3F	77	?
58	3A	72	60	3C	74	<
59	3B	73	62	3E	76	>
60	3C	74	64	40	100	@
61	3D	75	92	5C	134	\
62	3E	76	94	5E	136	^
63	3F	77	59	3B	73	;

0.3 Data section

The data section consists of one or more *blocks* aligned to 64-bit word boundaries. Each block is, in turn, comprised of one or more *cycles*. Each cycle contains one or more set of *parameter values*. Each set of parameter values contains one or more values; the number of values contained in each set is determined by the (relative) sample rate of each parameter.

The offset, in multiples of 8-bit bytes, from the start of the file to the data section is computed as

$$\text{offset}_{\text{data}} = 8 \left\lceil \frac{100 \times (11 + N) \times 6}{64} \right\rceil,$$

where N is the number of parameters in the file; i.e., the offset is in multiples of 64-bit words (presumably owing to the Cray-1's 64-bit word size). (The value 11 appears as there are always 11 text lines which precede the list of parameters.) For example, for the PHOENIX-78 dataset, $N = 67$, so the data section begins at the offset

$$\text{offset}_{\text{data}} = 8 \left\lceil \frac{100 \times (11 + 67) \times 6}{64} \right\rceil = 5856.$$

The stride between blocks, in multiples of 8-bit bytes, is

$$\text{stride} = 8 \left\lceil \frac{\text{cycles per block} \times \text{samples per cycle}}{64} \right\rceil,$$

unless the product (cycles per block × samples per cycle) is evenly divisible by 64, in which case the stride becomes

$$\text{stride} = 8 \left\lceil \frac{\text{cycles per block} \times \text{samples per cycle}}{64} \right\rceil + 1.$$

I.e., in this scenario (and in this scenario only), an extra zero 64-bit word is added as padding between consecutive blocks. Both the number of samples per cycle and the number of cycles per block are specified in the header. The number of samples per cycle may also be computed as:

$$\text{samples per cycle} = \sum_{i=0}^{N} (\text{sample rate for parameter } i),$$

where N is again the number of parameters, and R_i is the sample rate for the ith parameter. For the example dataset, there are 6 parameters with a sample rate of 1 (parameters 1 through 6), and 61 parameters which have a sample rate of 20 (parameters 7 through 67), so

$$\text{samples per cycle} = 6 \times 1 + 61 \times 20 = 1226.$$

Note that this equals the "samples/program cycle" figure given in the header.

Recall that each block is aligned to 64-bit word boundaries, but individual cycle and parameter values are not necessarily. For this reason, it is easiest to read entire blocks at time, then use `gbytes` to decode the data. Likewise, the data section may also be separated from the plain-text header by some amount of zero padding; for this reason, it is easiest to read the data section by seeking to the start of the data section relative to the start of the file, then reading one cycle's worth of data at a time until end-of-file (EOF) is reached.

Within each cycle of data, each parameter appears in order, such that if a parameter has a sample rate (in multiples of the cycle frequency) greater than 1, then there will appear that many instances of data for that parameter. E.g., if parameter 1 has a sample rate of 2, parameter 2 has a sample rate of 1, and parameter 3 has a sample rate of 5, one would see two parameter 1 values followed by one parameter 2 value, and followed up by five parameter 3 values. This organization of the data is illustrated in fig. 1.

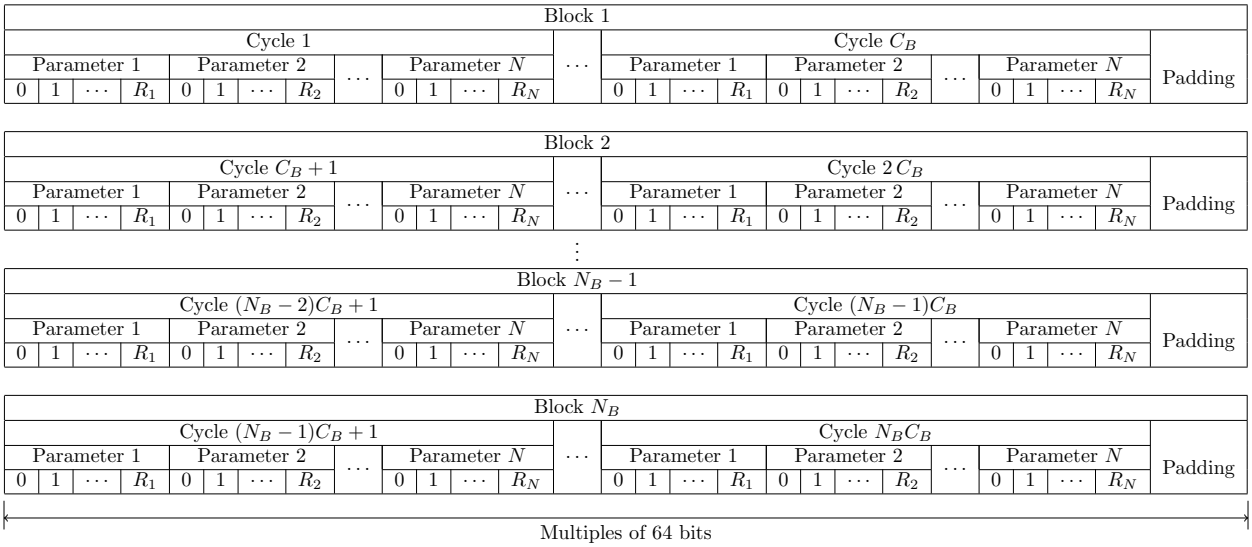

Figure 1: Data organization. N_B is the number of blocks contained in the file; C_B is the number of cycles per block. Note that there is no padding between consecutive cycles or parameters; only between adjacent blocks.

0.4 Implementation Notes

0.4.1 Unpacking 6-bit words

The following snippet of C code may be used to decode the 6-bit header data. This has been tested to work on the x86-64 platform with gcc 4.4.7.

```
void decode(uint8_t *in_buffer, char *out_buffer, size_t length)
{
   uint32_t buf[3];
   for (j = 0, i = 0; i < length; i += 12 /* 3*32/8 */, j += 16) {
      buf[0] = (((uint32_t) in_buffer[i+ 0]) << 24) |
               (((uint32_t) in_buffer[i+ 1]) << 16) |
               (((uint32_t) in_buffer[i+ 2]) <<  8) |
               (((uint32_t) in_buffer[i+ 3]) <<  0);
      buf[1] = (((uint32_t) in_buffer[i+ 4]) << 24) |
               (((uint32_t) in_buffer[i+ 5]) << 16) |
               (((uint32_t) in_buffer[i+ 6]) <<  8) |
               (((uint32_t) in_buffer[i+ 7]) <<  0);
      buf[2] = (((uint32_t) in_buffer[i+ 8]) << 24) |
               (((uint32_t) in_buffer[i+ 9]) << 16) |
               (((uint32_t) in_buffer[i+10]) <<  8) |
               (((uint32_t) in_buffer[i+11]) <<  0);
      out_buffer[j+15] = buf[2] & 0x3F; buf[2] >>= 6;
      out_buffer[j+14] = buf[2] & 0x3F; buf[2] >>= 6;
      out_buffer[j+13] = buf[2] & 0x3F; buf[2] >>= 6;
      out_buffer[j+12] = buf[2] & 0x3F; buf[2] >>= 6;
      out_buffer[j+11] = buf[2] & 0x3F; buf[2] >>= 6;
      out_buffer[j+10] = (buf[2] & 0x3) | ((buf[1] & 0xF) << 2); buf[1] >>= 4;
      out_buffer[j+ 9] = buf[1] & 0x3F; buf[1] >>= 6;
      out_buffer[j+ 8] = buf[1] & 0x3F; buf[1] >>= 6;
      out_buffer[j+ 7] = buf[1] & 0x3F; buf[1] >>= 6;
      out_buffer[j+ 6] = buf[1] & 0x3F; buf[1] >>= 6;
      out_buffer[j+ 5] = (buf[1] & 0xF) | ((buf[0] & 0x3) << 4); buf[0] >>= 2;
      out_buffer[j+ 4] = buf[0] & 0x3F; buf[0] >>= 6;
      out_buffer[j+ 3] = buf[0] & 0x3F; buf[0] >>= 6;
      out_buffer[j+ 2] = buf[0] & 0x3F; buf[0] >>= 6;
      out_buffer[j+ 1] = buf[0] & 0x3F; buf[0] >>= 6;
      out_buffer[j+ 0] = buf[0] & 0x3F; buf[0] >>= 6;
   }
}
```

Alternatively, the `gbytes` routine may be used, which is available for a variety of platforms in a variety of languages.

0.4.2 Ceiling of division

It is useful to define a macro to round up integer division (as this operation is required to compute the data offset and stride) such as the following:

```
#define DIV_CEIL(n,d) (((n)-1)/(d)+1)
```

0.5 Using the converter

The syntax of the converter is straightforward: one specifies the path to the input file, and a path to the desired output file.

```
$ genpro2nc INFILE OUTFILE
```

For example, starting with a COS-blocked file `G50222C`, we obtain a NetCDF file like so:

```
$ cosconvert -b -e bin G50222C
$ genpro2nc G50222C.bin G50222C.nc
```

(The `-e` switch to `cosconvert` tells `cosconvert` to create an un-COS-blocked file with the extension which immediately follows the flag; here, `.bin`.)

0.5.1 Output

`genpro2nc` generates an unlimited dimension, `Time`, which is the first dimension used by every array that is generated in the file. A number of global attributes are added (see sec. 0.6 information on changing these):

- `institution` – Set to "NCAR Research Aviation Facility".
- `Address` – The address of the institution, set to "P.O. Box 3000, Boulder, CO 80307-3000"
- `creator_url` – The web URL of the creating institution, set to "http://www.eol.ucar.edu".
- `ConventionsURL` – Set to "http://www.eol.ucar.edu/raf/Software/netCDF.html".
- `Platform` – The tail number of the aircraft used as a flight platform.
- `time_coverage_start`, `time_coverage_end` – formatted as `YYYY-MM-DDTHH:MM:SS +0000` (e.g., `1978-11-17T18:02:19 +0000`).
- `TimeInterval` – The range of time covered by the file, formatted as `HH:MM:SS-HH:MM:SS` (e.g., 15:53:15-18:02:19).
- `FlightDate` – The date the flight took place, formatted as `DD/MM/YYYY` (e.g., 11/17/1978).
- `geospatial_lat_min`, `geospatial_lat_max`, `geospatial_lon_min`, `geospatial_lon_max` – Minimum and maximum latitude/longitude values contained in the file.

Every variable has at least two attributes:

- `long_name` – a human readable description of the variable, and
- `units` – the units of the variable.

Every variable (except `Time`) also have these attributes:

- `SampledRate` – The sample rate of the array.
- `actual_range` – A vector of two floating point values indicating the minimum and maximum values found in the array.

Variables which are sampled at a rate higher than the cycle rate of the source GENPRO-I file are stored as two-dimensional arrays; the first dimension is, as always, `Time`, and the second dimension indicates the sample rate of the variable and is named of the form `sps#` (e.g., `sps5`).

0.6 Extending & Modifying the Converter

`genpro2nc` works in three steps:

1. Read the GENPRO-1 data without altering it in any way (`gp1_read()`).
2. Apply *rules* to alter how the data will be output (`rule_applyAll()`).
3. Write the file to NetCDF (`gp1_write_nc()`).

There should be no need to modify either `gp1_read` or `gp1_write_nc`. Instead, one should be able to accomplish any desired change in the output by editing the rules defined in `rules.cpp`. These rules are contained in the `rules` array, which is an array of `Rule` structures.

For example, if one wanted to add a constant (i.e., unchanging, fixed) global attribute to the output, one would first define the attribute:

```
Attribute creatorGlobalAttr = {
   (char*) "creator",                    /* name */
   kAttrTypeText,                        /* type */
   (char*) "Your Organization Name Here" /* value */
};
```

Next, we define an array of *applicator data*. Each entry in the array (of type `RuleApplicatorData`) contains an *applicator* (a callback which effects some change on the output) and data that will be passed the applicator as an argument. To add a constant global attribute, we will use the `rule_addGlobalAttr` applicator, which requires an `Attribute` as an argument:

```
RuleApplicatorData creatorGlobalAttrApplicatorData[] = {
    { rule_addGlobalAttr, &creatorGlobalAttr }
};
```

(Note that this must be declared as an array even if there is only one attribute.)

Finally, this rule would need to be added to the global `rules` array:

```
Rule rules[] = {
    // Add global attributes which should always be present
    {
        NULL,
        rule_alwaysApplyGlobal,
        creatorGlobalAttrApplicatorData, 1 /* number of applicators */
    },
    // Other rules follow
    // ...
};
```

We used the `rule_alwaysApplyGlobal` rule, which always executes its applicators once for the entire file.