Non-observation RINEX compression

Juraj BEZRUČKA[1]

[1] Department of Theoretical Geodesy, Faculty of Civil Engineering
Slovak University of Technology, Radlinského 11, 811 68 Bratislava, Slovak Republic
e-mail: juraj.bezrucka@stuba.sk

Abstract: RINEX format is a result of an effort to standardize exchanging GNSS related data in human readable receiver independent format. Besides observation data, RINEX is also *de facto* standard for exchanging navigation message or meteorological data. The paper describes simple compression methods for navigation messages and meteorological data files. Both formats have their specific features, therefore different methods were used. Followed by standard compression the algorithms reduce the file size by more than 20 percent compared to standard compression. Thanks to the properties of weather and regular parameter observations it allows to reduce the file size of meteorological RINEX to a mere fraction of its original size.

Key words: RINEX, data compression, navigation message, meteorological data

1. Introduction

File compression is nowadays daily routine for many people. There are plenty of standard compression methods, among the most popular are ZIP, Z, GZ, 7Z, RAR, ARJ and others. Most of the methods are based on the LZ technique. In their famous papers *Ziv and Lempel (1977)* and *(1978)* introduced a new method that searched for repeating sequences in text and replaced and encoded these into few bits. The method has been modified several times and improved by the original authors and also by others: LZW (*Welch, 1984*), LZSS (*Storer and Szymanski, 1982*), LZMA/LZMA2 (*Cormack and Horspool, 1987*).

The most common method of RINEX data compression is standard LINUX *compress*, sometimes denoted as Z (because compressed file has an extension .Z). This compression method is considered as *standard* compression in this paper. It is also based on LZ algorithm.

Contrary to standard sequence based compression there may be file specific compressions. These compressions use the specific features of data stored in them to compress them on other sequence based principles. One example for all can be CRINEX or Compressed RINEX format (*Hatanaka, 1996*). Y. Hatanaka has developed a very effective and powerful algorithm for compressing observation data. RNX2CRX compression program takes advantage of the fact that data concerning single satellite are similar and change slightly if observed regularly. Therefore instead of parameter values, the 3rd order differences are used, for more details see *Hatanaka (2008)*. Unchanged values are omitted and using this approach the observation files can be compressed to approximately third of original size. The resulting text file is still suitable to be compressed with standard LZW techniques.

Although the observation data files use by far the largest amount of data storage, the other RINEX types are being stored as well and these are usually only compressed by standard means. The navigation messages as well as meteorological data files are usually rather small, typically several tens to hundreds of Kilobytes. The need to compress the data is therefore not as urgent as in the case of observation files. On the other hand, smaller files take less storage space and less time to send via network. This led to testing the possibilities of compression of both file types. The tests have been examined on widely used RINEX format v2.11 (*Gurtner, 2002*) with intention to upgrade the algorithm to format RINEX 3 if compression proves to be effective. It is expected that the upgrade to an up-to-date RINEX version would require only minor changes while the principles of the compression would stay unchanged.

2. Navigation message

Navigation message is a list of values needed to describe the satellites orbit. These are further used to determine satellite precise position at the specific epoch. The number and type of parameters are specific for each satellite system. Like all other RINEX files, the navigation RINEX consists of a header section and the data section (*Gurtner and Estey, 2009*).

The header stores required and additional information. The 80-character line length is divided to two parts (60 and 20 characters) that defines the parameter value and parameter description. There may be more parameters in a single line.

The data section follows right after "END OF HEADER" caption. Each line is divided into five columns. The first one is only 3 characters long and it shows the satellite number in the first line or is left empty in every other line concerning a single satellite. The following 19 characters of a first line is reserved for time information. The rest of data is stored in 19-character long columns in scientific floating point format. Although the format specifications (2.10 or 3.01) require numbers to be written in D19.12 format, it is not strictly defined if leading zero or leading non-zero should be used. Some navigation messages use floating point as a first character. These differences can be easily handled when compressing data, but they are always decompressed to the format with non-leading zero, as it stores maximum digits (13). The compression therefore cannot be considered 100% lossless, although the values keep unchanged.

Unlike in observation RINEX, most of the data in navigation message changes dramatically, as there are different satellites in different positions with respect to the Earth in different time epochs and different data type. Therefore the differences cannot be used. However, there are several facts that can be utilized when intending to compress data.

2.1. Redundant characters and data

The only redundant character in header is usually an empty space. If the sequence of empty spaces exceeds 3, it can be easily substituted with $DD, where DD is an integer describing the length of the space sequence. If a half of the 60-character long space is empty, the reduction is quite effective. However a header section is rather short and if we intend to further compress it with standard tools, it may not lead to considerable results. This compression is a recommended option but can be easily turned off without dramatic loss of reduction.

The part where we expect most of compression is a data part. First of all we read the first line of data section and define the reference time. Many of the navigation messages are sorted timewise from oldest to latest data. Therefore if we keep the reference record and only use time deltas (in seconds) the most of reserved 19-character long room can be saved.

The current version of compression tool swaps the time information and satellite number. This has been chosen as a suitable solution for compatibility with CMET tool which also displays time deltas at the first place. If there is no time delta (i.e. the epoch of data is the same as a reference) it is left empty and the satellite number follows.

When we examine the rest of data, in either GPS or GLONASS navigation messages, we can easily identify redundant characters repeating in every single value. These are a *dot* (floating point) and an *exponent marker* (in RINEX 2.11 usually D, later E). Skipping these characters would save 8 out of 80 characters of a line (10%). We save 3 more characters in most of the lines if we skip first 3 characters as they are empty in every data line except the first one. However, this gain is balanced with 3-space loss when separating spaces are wedged between the values. Thanks to the separating spaces, the positive values do not have to be signed.

Thus we have removed all redundant characters occurring in every number. However, there can be plenty of more. First of all, the leading zeros in exponents should be removed. As the most of the exponents are within range $< -9; 9 >$ we can save some digits. If we further omit the zero exponents completely and left only sign ("+" or "−") if the exponent is +1 or −1, we can save another valuable characters.

2.2. Character groups replacement

If a value of a parameter is 0, it takes 19 characters of storage in the RINEX format. Let us leave out the value completely. If a value is expected and nothing is found, the value is 0. This can save us all 19 characters except separating space. Thus one or more empty spaces might occur following each other. We can further introduce the replacement: "_" for two spaces, "=" for three and "|" for four empty spaces and thus compress more zero values to a single character.

Hatanaka algorithm takes advantage of periodically repeated observation of the same satellites. The observed values therefore change only of a small amount and can be counted using differences. As the orbits of the observed satellites are very smooth, the differences can be determined very precisely and therefore the remaining deltas are small and take much less bytes. The NAV files are different. They provide us with the complete information for

Table 1. Suggested conversion table for half byte conversion. "0x" prefix denotes hexadecimal number

0x0	***0x1***	***0x2***	***0x3***	***0x4***	***0x5***	***0x6***	***0x7***
0	1	2	3	4	5	6	7
0x8	***0x9***	***0xA***	***0xB***	***0xC***	***0xD***	***0xE***	***0xF***
8	9	+	–	(space)	&	(empty)	(reserved)

each satellite. The satellite numbers may repeat but usually they are used only a single time as it is also recommended in (*Gurtner and Estey, 2009*).

Despite aforementioned, several parameters may have the same value for more or even all satellites (GPS week for instance). These values, after once defined, can be replaced by a single character ("!") until different value occurs for this parameter.

The occurrence of the very same values is rather rare in NAV files. However, the repeating exponents are not rare at all. As they describe similar phenomena, the values are usually very similar in the terms of exponents. Therefore by replacing all repeating exponents (except −1, 0 and 1 for obvious reasons) by "?" we can save some more space.

Using the procedure above we can compress NAV RINEX to approximately a half of its original size. As mostly text replacement methods were used, this file can still be further compressed using standard methods. The advantage of this step-by-step compression is that almost every step can by skipped if a user wishes to keep more than required minimum of information. The results of this compression, further referred as CNAV, are shown in Table 3 and compared with other compression methods.

2.3. Half byte compression

When closely inspecting RINEX files it is obvious that the data part consists mostly of digits 0-9, empty spaces, + and – signs and/or few other characters. If we only need 16 characters to cover all the needs of data section, we can split 8 bits of a byte (character) to two parts and thus compress two characters to a single one.

Using the CNAV compression described in the previous section, we can convert a number "+5.258714651000D+06" to "5258714651+6". It has shrunk from 19 to 12 characters. However if we further use the conversion table (Table 1) it translates to 6-character long record: "RXqFQ|" (as byte value 0x52 represent character value "R", 0x58 represents "X" etc.).

If odd number of characters is used, the empty sign 0xE is used to complement the couple. As the new line character is *0x13* in hexadecimal, to avoid mismatch (translating a new line to decimal 13), different new line character must have been chosen. Thanks to reserved half byte 0xF, there is a range of free bytes that can be used from *F0* through *FF*. For a newline *0xFF* is used. There are also special bytes for beginning of file, beginning and end of an uncompressed section.

This compression reduces the size substantially. However, as the output is not human readable and contains many more characters, fewer patterns appear there and therefore standard compression methods are not as effective as usually. The tests of various RINEX files proved that the most effective (and recommended) technique is a CNAV compression followed by standard *compress*. The only exception is compressing very large navigational RINEX files. In such case, the half byte compression is slightly more effective, although the storage reduction is usually negligible.

The half byte compression has also been tested on Hatanaka compressed observation RINEX. CRINEX data section only contains characters listed in Table 1 (except plus sign), therefore it is possible to apply this method. The files were compressed by standard compress function after half byte conversion. Although the half byte algorithm reduces the size of compressed RINEX to roughly a half of the original size, further standard compression usually ends up with file size slightly larger than Z compressed CRINEX. See Table 3 for results.

2.4. Binary compression

The half byte compression makes use of the fact that most of the data in RINEX files are digits. In fact, all characters are ordered in a specific way to represent certain numbers, even minus sign or exponent characters (this can be D or E, the former is more frequent) have their meanings. This allows to translate and write down every number in its binary form.

Each number can be expressed as floating point number and an exponent. Ignoring floating point we convert this decimal number to binary as well as exponent. One byte offers range 256 values. If first bit is reserved for sign, it offers range (−127:127). Note that standard range is −128:127, one byte (10000000 or decimal −0) thus stays reserved for special purposes. Such range exceeds by far the needs of exponent values in RINEX (these are usually in magnitudes up to 12 in absolute values). Therefore we only need one byte or none for exponent part.

If 127, a 3-digit long number, is maximum possible value that can be written in binary using a single byte, how many bytes are needed for 13-digit long numbers in RINEX?

Table 2 shows number of bytes required to write certain maximum value. As the maximum number of decimal places used in RINEX file is 13, it is obvious that the maximum length of a number would take $6 + 1$ characters (6 bytes for "base" is sufficient and 1 byte for exponent). This may lead to reserving $4 \times 7 = 28$ characters in each line for data, which is a rather good compression from original 79 characters. Many numbers require less than 6+1 bytes in their binary form, therefore many empty bytes (0x0) may appear which may lead to better results when further compressing with standard tools (.Z). This compression has been tested and further denoted as NAVBIN4.

Table 2. The largest number when using specific number of bytes. If there is a possibility to define negative sign elsewhere, the range is doubled

# bytes used	1	2	3	4	5	6
+/– range	127	32767	8.388e+6	2.147e+9	5.498e+11	1.407e+14

However, if we suppose that at least four bytes can be saved when not filling reserved bytes, it is worth to use another approach. Let us use the first byte to define number of values in line (usually $n = 4$) and following $n/2$ bytes defining which bytes are reserved for which numbers. As we always use less than 8 bytes for a base number and at most 1 byte for an exponent, 3 bits can be used for a base and 1 for the exponents. If odd number of values are used (e.g. in first line of data), 4 free bits 0000 are added to complete a byte. This definition of bytes are followed by values in binary. As the positions are strictly given, no division marks are needed between numbers.

Now we will examine the conversion on the same number as in the case of half byte compression. The number "+5.258714651000D+06" is first divided into two parts (5258714651,6) and both numbers converted to binary: (0000000100111001011100011001111000011011, 00000110). It will thus need 5+1 bytes to write it down and half of a byte to define the position.

Let us see what happens if we compress the whole line:

4.100000000000D+01−5.656250000000D+00 5.263790686582D-09 1.553898 170302D+00

3 bytes are used to define the number of values and positions. The values of a line can be stored in (1+1) + (3+0) + (6+1) + (6+0) bytes. 79 characters are thus compressed to 21 bytes.

2.5. Comparison of various methods for navigation message compression

The results of testing various navigation message compression methods are available in Table 3C. The NAV files for stations ALIC (Australia), GANP (Slovakia) and MIKL (Ukraine) are compared in the table, although several other stations were tested as well.

Table 3 compares the performances of various compression methods. The navigation messages of various satellite systems and various stations were used. It is obvious that the compression itself is usually less effective than standard LZW compression. Therefore all comparisons and conclusions should be done when the files are further LZW compressed. There are several conclusions based on the results in Table 3 and other tests:

1. The half byte compression is less effective than LZW compressed CNAV in most cases.
2. The performance of NAV2BIN is better or comparable with NAV4BIN.
3. NAV2BIN does not have to be the most effective way of compressing data, CNAV may perform better when not compressing very small or very large files.
4. Aforementioned is true regardless of whether compressing GLONASS or NAVSTAR files.

Table 3. Performance of various compression methods. .n or .g denotes navigation RINEX (NAVSTAR or GLONASS, respectively), Z denotes LINUX Z compression (compress), C CNAV compression, H additional half byte compression after CNAV compression, B4 NAV4BIN and B NAV2BIN compression methods

Stn ALIC	**g**	**g.Z**	**g.C**	**g.C.Z**	**g.H**	**g.H.Z**	**g.B4**	**g.B4.Z**	**g.B**	**g.B.Z**
Bytes	*14018*	*4873*	*7149*	*3761*	*4023*	*3932*	*5397*	*3742*	*4210*	*3569*
g %	100.0	34.8	51.0	26.8	28.7	28.1	38.5	26.7	30.0	25.5
g.Z %		100.0	146.7	**77.2**	82.6	80.7	110.8	76.8	86.4	**73.2**
g.C %			100.0	52.6	56.3	55.0	75.5	52.3	58.9	50.0
g.C.Z %				100.0	107.0	104.5	143.5	99.5	111.9	94.9
Stn GANP	**n**	**n.Z**	**n.C**	**n.C.Z**	**n.H**	**n.H.Z**	**n.B4**	**n.B4.Z**	**n.B**	**n.B.Z**
bytes	*125113*	*37293*	*63997*	*29752*	*35270*	*32512*	*47220*	*33069*	*36856*	*33107*
n %	100.0	29.8	51.1	23.8	28.2	26.0	37.7	26.4	29.5	26.5
n.Z %		100.0	171.6	**79.8**	94.6	87.2	126.6	88.7	98.8	**88.8**
n.C %			100.0	46.5	55.1	50.8	73.8	51.7	57.6	51.7
n.C.Z %				100.0	118.5	109.3	158.7	111.1	123.9	111.3
Stn MIKL	**n**	**n.Z**	**n.C**	**n.C.Z**	**n.H**	**n.H.Z**	**n.B4**	**n.B4.Z**	**n.B**	**n.B.Z**
bytes	*244406*	*66899*	*127322*	*54219*	*70276*	*54273*	*93731*	*53415*	*73136*	*51232*
n %	100.0	27.4	52.1	22.2	28.8	22.2	38.4	21.9	29.9	21.0
n.Z %		100.0	190.32	**81.0**	105.0	81.1	140.1	79.8	109.3	**76.6**
n.C %			100.0	42.6	55.2	42.6	73.6	41.9	57.4	40.2
n.C.Z %				100.0	129.6	100.1	172.9	98.5	134.9	94.5

3. METEO RINEX

METEO RINEX consists, as a navigation message, of a header part and a data part. The header follows the same specifications like the navigation message. It can but does not have to be compressed using the empty space replacement described in CNAV section.

The data part structure is different than in NAV RINEX. There is an epoch information in the first 18 characters. The following characters are used for meteorological parameters. Each of the parameters takes exactly 7 characters (format %7.1f) with leading empty spaces. No empty spaces are added between parameters. If more than 8 parameters are used, the following lines can be used according to *Gurtner and Estey (2009)*. However, only files with maximum of 7 parameters were used when testing the algorithm.

The time delta between following epochs is usually a constant. Therefore if we set up a reference epoch in the first data line and the time delta in the second, the time info can be omitted totally in the rest of the file, unless there are missing epochs. In such a case, the reference time and delta must be redefined again. Using this approach we save 18 characters of each line besides the first two.

The RINEX specification reserves exactly 1 decimal place for each of the parameters. Therefore when text compressing meteorological data, these are multiplied by 10 and used as integers instead of floating point numbers. This saves a dot in each parameter.

The weather is an ideal phenomenon to describe by differences. Especially when short intervals are used, the meteorological parameters change subtly or not at all. There are three possible changes: no change signalized by a character "0", increase characterized by "+" and its value or decrease characterized by "−" followed by its absolute value. The zero differences can be as well substituted with empty spaces and further replaced by special signs for space substitution described in CNAV section. The rest of the values are differences. They are separated by their sign, therefore no additional character (such as empty space) must be added between the values. If $+1$ or -1 is to be used (i.e. when the parameter value changes by 0.1), the digit can be omitted.

As the meteorological data changes only slightly, there are many epochs when no change appears in more parameters. This leads to conversion where no digits are used and therefore the half byte conversion does not even have to be considered as it compresses less effectively than standard algorithm described above.

Using this simple routine the meteorological RINEX file can be compressed to significantly smaller size than using standard LZW compression and still be fit for further standard compression. The tests were examined on three meteorological RINEX files from different stations with various record rate BOR1 in Poland (10 minutes), ALIC in Australia (30 second rate) and TITZ in Germany (10 seconds). For results see Table 4.

Table 4 shows that CMET compression alone is more effective than LZW compression. The larger files are compressed better because the shorter interval means (besides more data) lesser changes in parameters and therefore better compression. As the CMET compressions rely fully on text manipulations (unlike the binary decomposition based compression methods), there are always many patterns that can be efficiently compressed by LZW. In Table 4 we can see the best result – the file compressed to 1.7% of its original size or 9.5% of the LZW compressed size. The compression rate in

Table 4. Performance of various compression methods: .m denotes uncompressed meteorological RINEX, Z denotes LZW (Z) compression and C CMET compression

Stn BOR1	**.m**	**.m.Z**	**.m.C**	**.m.C.Z**
bytes	*7416*	*2048*	*1540*	*866*
.m %	100.0	27.6	20.8	11.7
.m.Z %		100.0	75.2	42.3
.m.C %			100.0	56.2
Stn ALIC	**.m**	**.m.Z**	**.m.C**	**.m.C.Z**
bytes	*197160*	*25094*	*14216*	*3956*
.m %	100.0	12.7	7.2	2.0
.m.Z %		100.0	56.7	15.8
.m.C %			100.0	27.8
Stn TITZ	**.m**	**.m.Z**	**.m.C**	**.m.C.Z**
bytes	*325121*	*57843*	*23587*	*5520*
.m %	100.0	17.8	7.3	1.7
.m.Z %		100.0	40.8	9.5
.m.C %			100.0	23.4

BOR1 station is worse because the interval between records is 10 minutes which means larger changes in weather parameters, which leads to worse compression as less values can be skipped. However, CMET compression is more efficient than LZW even in this case.

4. Conclusions

The simple modular compression of non-observation RINEX files has been proposed and tested. It is based on text replacements and leaving out unnecessary information. The compression provides several steps to compress numeric data stored in the files. If needed for any reason, the most of the steps can be skipped according to the required level of compression. Because the nature of the navigation message and meteorological RINEX is different, various methods were used for compression of navigation message (CNAV) and meteorological RINEX (CMET).

Two other compression methods based on binary decomposition have also been proposed and tested. The resulting files were further compressed by standard LZW (Z) compression which led to smaller files. The NAV2BIN compression is a good tool for compressing very small or very large NAV files. However, for standard file sizes it is recommended to use CNAV method followed by LZW compression.

The resulting file sizes have been compared to traditionally compressed non-observation RINEX file. The navigation messages can be compressed to approximately 80% or less of the original compressed file or to 15–30% of the original navigation message size. The meteorological data, thanks to the wheather properties, can be compressed to 10–40% of compressed file (the larger the file, the better the compression) or to 2–10% of original RINEX size.

Acknowledgments. This publication is the result of project implementation: National Centre of Earth's Surface Deformation Diagnostic in the area of Slovakia, ITMS 26220220108 supported by the Research and Development Operational Program funded by the ERDF.

References

Cormack G., Horspool N., 1987: Data Compression using Dynamic Markov Modelling. Computer Journal, 30:6 (December 1987).

Gurtner W., 2002: RINEX: The Receiver Independent Exchange Format Version 2.10. `ftp://igscb.jpl.nasa.gov/igscb/data/format/rinex210.txt`.

Gurtner W., Estey L., 2009: RINEX. The Receiver Independent Exchange Format. Version 3.01. `http://igscb.jpl.nasa.gov/igscb/data/format/rinex301.pdf`.

Hatanaka Y., 1996: A RINEX Compression Format and Tools. Proceedings of ION GPS-96, September 17-20, 1996, 177–183.

Hatanaka Y., 2008: A Compression Format and Tools for GNSS Observation Data. Bulletin of the Geographical Survey Institute, Tsukuba, Japan, **55**, 21–30.

Storer J., Szymanski T., 1982: Data compression via textual substitution. Journal of the ACM, 928–951.

Welch T., 1984: A Technique for High-Performance Data Compression, IEEE Computer, **17**, 6, 8–19.

Ziv J., Lempel A., 1977: A universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory, **23**, 3, 337–343.

Ziv J., Lempel A., 1978: Compression of Individual Sequences via Variable-Rate Coding. IEEE Transactions on Information Theory, **24**, 5, 530–536.

Appendix 1: Comparison of uncompressed NAV RINEX and CNAV RINEX

NAV RINEX

```
32 13  6 26  0  0  0.0-5.590058863163D-04 2.273736754432D-13 0.000000000000D+00
    4.100000000000D+01-5.656250000000D+00 5.263790686582D-09 1.553898170302D+00
   -2.458691596985D-07 1.156179106329D-02 4.824250936508D-06 5.153652671814D+03
    2.592000000000D+05 5.960464477539D-08-8.333805729374D-01 1.844018697739D-07
    9.488829745675D-01 2.770000000000D+02-3.444540169210D-01-8.366777081277D-09
   -4.457328522767D-10 1.000000000000D+00 1.746000000000D+03 0.000000000000D+00
    2.000000000000D+00 0.000000000000D+00-2.793967723846D-09 4.100000000000D+01
    2.520180000000D+05 4.000000000000D+00
20 13  6 26  0  0  0.0 1.057083718479D-04 2.160049916711D-12 0.000000000000D+00
    9.400000000000D+01 4.718750000000D+00 5.894531245172D-09-4.280157688719D-01
    2.458691596985D-07 5.380079732276D-03 4.602596163750D-06 5.153710874557D+03
    2.592000000000D+05-5.960464477539D-08-9.737472364124D-01 3.539025783539D-08
    9.276153173013D-01 2.711562500000D+02 1.294193023854D+00-8.864654962747D-09
   -1.503634060966D-10 1.000000000000D+00 1.746000000000D+03 0.000000000000D+00
    2.000000000000D+00 0.000000000000D+00-7.916241884232D-09 9.400000000000D+01
    2.529780000000D+05 4.000000000000D+00
```

CNAV

```
&&20130626
& 32 -5590058863163-4 2273736754432-13
41+ -565625 5263790686582-9 1553898170302
-2458691596985-7 1156179106329-2 4824250936508-6 5153652671814+3
2592+5 5960464477539-8 -8333805729374- 1844018697739-7
9488829745675- 277+2 -344454016921- -8366777081277-9
-4457328522767-10 1 1746+3
2_-2793967723846-9 41+
252018+5 4
& 20 1057083718479? 2160049916711-12
94+ 471875 5894531245172? -4280157688719-
2458691596985? 5380079732276-3 460259616375? 5153710874557?
! -5960464477539? -9737472364124- 3539025783539-8
9276153173013- 27115625? 1294193023854 -8864654962747?
-1503634060966? ! !
!_-7916241884232? 94+
252978? !
```

Appendix 2: Comparison of uncompressed METEO RINEX and CMET RINEX

METEO RINEX

```
 13 07 01 00 00 00 1003.1   11.6   98.9    0.0
 13 07 01 00 10 00 1002.7   11.6   99.3    0.0
 13 07 01 00 20 00 1002.7   11.6   99.4    0.0
 13 07 01 00 30 00 1002.9   11.7   99.9    0.0
 13 07 01 00 40 00 1002.7   11.8   99.9    0.0
 13 07 01 00 50 00 1002.7   11.9   99.9    0.0
 13 07 01 01 00 00 1002.7   12.0   99.9    0.0
 13 07 01 01 10 00 1002.3   12.0   99.9    0.0
 13 07 01 01 20 00 1002.7   12.0   99.9    0.0
 13 07 01 01 30 00 1002.5   12.1   99.9    0.0
```

CMET

```
&&20130701000000 10031 116 989
&600 -4 +4
_+
+2++5
-2+_
 +_
 +_
-4=
+4=
-2+_
```