# CIS File Format (Preliminary)

This is an attempt to describe the current .CIS file format that is used with the roll scanner being developed by the IAMMP group. This information came from reading the Q-Basic software that was provided by Richard Stibbons.

NOTE: The format of the .CIS file may be changed at any time making this document obsolete.

The original intent of this document is to give other people an understanding about what is contained in the .CIS files. It is not intended to be a specification of this interface (but it can be). I have not talked to Richard Stibbons, Warren Trachtman or Gene Gerety about generating this document. It would be good to get their input about my interpretation of this interface. Some things might be described wrong but any type of input will be good. At least there is something to discuss now.

I believe that this interface should be standardized and documented thoroughly. So many of the scanned images will be generated in the future, it would be best that this interface be well defined and flexible enough so different software packages can be written around the same file structure. At this point I don't think that either of these have happened so this is what has prompted my writing this document. I am not a writer but this is a start in the right direction.

Note: Some of the language used in this document is specific to the programming environment and my not be understood unless you are a programmer.

## Definitions

**BYTE** – 8 bits that are typically considered as one unit and are used to represent a single character such as a number, letter, or a special control character. It can hold a value from 0-255 (0x00 – 0xFF).

**WORD** – 2 BYTES. When combined can hold a value from 0-65535 (0x0000 – 0xFFFF). Because the PC uses the INTEL compatible processor, the WORD values are stored in reverse order. That is Low order BYTE first and high order BYTE last. This makes reading the file awkward as a human.

**DOUBLE WORD –** 4 BYTES (values reversed like described in a WORD).

**HEX** (Hexadecimal) Number – This is a numbering system that represents a value found in a BYTE or WORD or DOUBLE WORD. Since the numbers are represented using binary 1's and 0's, it is easier to describe the value as **0x**35 rather than binary 0011 0101. The **0x** preceding the number indicates that the value represents a hex value. Each BYTE which contains 8 bits, are broken into two (4 bit) hexadecimal numbers. Each HEX number can be represented by the following values: 0 1 2 3 4 5 6 7 8 9 A B C D E F

## Basic Format (Header and Scanned Data)

The following table describes the information found in the .CIS files.

| Position in File (Decimal) | Value Name | Value Type | Description of the field |
|---|---|---|---|
| 0-39 | Roll Description | BYTE (40 Characters long) | Text describing the roll being scanned. The values are ASCII characters between the value of 0x20 and 0x7e. Any characters not used in this field are defined as a SPACE (0x20). A special character of 0x00 could be used to terminate the field prematurely but I don't know if this would cause problems or not. |
| 40-41 | Scanner Width | WORD | Number of bytes that can be read from the scanner for one scan line. The default for this value is 2432 (0x0980) |
| 42-43 | Reserved1 | WORD | Spare value not used any more and should be ignored. This was the old Obsolete Hangover value (whatever that is). |
| 44-45 | Tempo | WORD | Tempo on the roll. If this value is 0, then a default value of 90 (0x005A) is used. |
| 46-47 | LPI | WORD | ?????? Lines Per Inch in the vertical axis. This describes the number of steps needed to move the paper roll or media 1 inch. The default for this value is 182 (0x00B6). I am not sure that this isn't the horizontal dot spacing of the CIS tube ???????. |
| 48-51 | Line Count | DOUBLE WORD | Number of scan lines in this .CIS file. |
| 52- End Of File | Scanner Data | WORD[] | This is a compressed version of the scanned data that the Contact Image Sensor saw for the entire scanned media or roll. Each scanned line is represented using multiple WORD's with a **Flag** WORD at the end indicating the encoder reading and the Q-Basic real time clock value. See **Scanner Data Format** section for more information. |

## Scanner Data Format

The Scanner data found in the .CIS file is found after the header information. It contains 1 or more WORD values that represent a light change as the data was being clocked out of the Contact Image Sensor. A light change is defined as a change from dark to light or light to dark. When this light change is detected, then a number is stored in the file indicating its relative position from the last change. This clocking and storing continues until all of the data has been clocked out of the CIS array and a final count is then stored in the file. You can determine the end of the scanned line by accumulating consecutive WORD values until it reaches the Scanner Width value.

After the entire scanned line is clocked out of the CIS sensor, a final **Flag** WORD is stored. This **Flag** WORD contains the current Encoder Driver value and the Real Time Clock value. The Encoder Driver value counts down from 0x000c to 0x0008 to 0x0004 to 0x0000 and then back to 0x000c as shown in the example. I don't know if this value always starts at 0x000c. It might depend on your hardware. As far as the Real Time Clock value, I didn't see this value change on a large file so I don't think this is working. You should be able to discard this **Flag** value unless you want to do extra verification if the file is corrupt.

Here is an example of 2 scanned lines.

| | | | | | |
|---|---|---|---|---|---|
| 0x0003 | 0x0006 | 0x0003 | 0x0001 | 0x0973 | 0x0008 (Flag) |
| 0x0001 | 0x00BA | 0x0810 | 0x00B5 | 0x0004 (Flag) | |

Line 1: 0x0003 + 0x0006 + 0x0003 + 0x0001 + 0x0973 = 0x980 (which is 2432 Decimal, scanner width)
Flag is 0x0008
Line 2: 0x0001 + 0x00BA + 0x0810 +0x00B5 = 0x980 (2432 Decimal) Flag is 0x0004

The next scanned line will follow immediately after the last flag WORD.

## File Format Checking

Basic checking should be done while reading the CIS files. This will help identify corrupt or bad files as they are being uploaded and downloaded from the Internet.

1. As the scan line is being read, the accumulated values should always equal the scanners width. This is what determines the end of the scanned line. If this accumulation ever goes over the scanners width, then one or more of the data values is corrupt and the processing can be stopped immediately.
2. As each scan line is read from the file, an accumulated line count should be kept. When you get to the end of the file you should check to see if the accumulated line count is the same as the line count found in the header. If it isn't, something is wrong in the file.
3. I do not suggest using the Flag byte for your processing. It looks like it is very dependent on the hardware being used and my not be the same from one scanner type to another.

## File Example of the .CIS data

The following is the example that was found in the SCANNER.ZIP file that R. Stibbons placed on the web page some time ago. It shows exactly how the bytes are stored in the 40057AO.CIS file. I only included the top 400 bytes of the file in this example. Notice that the WORD defined values are all reversed or swapped. The fields are define as:

| | |
|---|---|
| Roll Description | = R Stibbons © 2000 02-07 (0x52, 0x20, 0x53, 0x74,…, 0x20, 0x20) |
| Scanner Width | = 0x0980 (2432 Decimal) |
| Reserved | = 0 (0x0000) |
| Tempo | = 0x0037 (55 decimal) |
| LPI | = 0x00B6 (182 decimal) |
| Line Count | = 0x0000792E (31022 total scan lines in file) |

```
52 20 53 74 69 62 62 6F 6E 73 20 28 63 29 20 32   R Stibbons (c) 2
30 30 30 20 30 32 2D 30 37 20 20 20 20 20 20 20   000 02-07
20 20 20 20 20 20 20 20 80 09 00 00 37 00 B6 00
2E 79 00 00 01 00 28 09 57 00 0C 00 03 00 06 00
03 00 01 00 73 09 08 00 80 09 04 00 01 00 BA 00
10 08 B5 00 00 00 02 00 B9 00 10 08 B5 00 0C 00
02 00 B9 00 10 08 B5 00 08 00 01 00 BA 00 10 08
B5 00 04 00 02 00 B9 00 11 08 B4 00 00 00 01 00
BA 00 10 08 B5 00 0C 00 02 00 B9 00 10 08 B5 00
08 00 02 00 BA 00 10 08 B4 00 04 00 02 00 BA 00
10 08 B4 00 00 00 01 00 BB 00 0F 08 B5 00 0C 00
02 00 B9 00 10 08 B5 00 08 00 02 00 B9 00 10 08
B5 00 04 00 02 00 B9 00 10 08 B5 00 00 00 02 00
B9 00 10 08 B5 00 0C 00 01 00 BA 00 0F 08 B6 00
08 00 02 00 B9 00 10 08 B5 00 04 00 02 00 B5 00
01 00 02 00 12 08 B4 00 00 00 01 00 BA 00 10 08
B5 00 0C 00 01 00 BA 00 10 08 B5 00 08 00 02 00
```

```
B9 00 10 08 B5 00 04 00 01 00 BA 00 10 08 B5 00
00 00 01 00 BA 00 10 08 B5 00 0C 00 02 00 B8 00
11 08 B5 00 08 00 02 00 B8 00 12 08 B4 00 04 00
01 00 BA 00 10 08 B5 00 00 00 01 00 B9 00 11 08
B5 00 0C 00 02 00 B9 00 10 08 B5 00 08 00 02 00
B8 00 11 08 B4 00 01 00 04 00 01 00 BA 00 10 08
B5 00 00 00 01 00 BA 00 10 08 B5 00 0C 00 02 00
B9 00 10 08 B5 00 08 00 01 00 BA 00 10 08 B5 00
```

The following table is the same file printed out in the INTEL WORD format so that the scan line values are more easily read. The software swapped the 2 BYTE values for you. I have used the "|" character to help separate the header fields and scan lines. Notice that the first 40 bytes of the description has every other byte reversed (ignore this).

```
2052    7453    6269    6f62    736e    2820    2963    3220
3030    2030    3230    302d    2037    2020    2020    2020
2020    2020    2020    2020  | 0980  | 0000  | 0037  | 00b6
792e    0000  | 0001    0928    0057    000c  | 0003    0006
0003    0001    0973    0008  | 0980    0004  | 0001    00ba
0810    00b5    0000  | 0002    00b9    0810    00b5    000c |
0002    00b9    0810    00b5    0008  | 0001    00ba    0810
00b5    0004  | 0002    00b9    0811    00b4    0000  | 0001
00ba    0810    00b5    000c  | 0002    00b9    0810    00b5
0008 | . . .
```