



# eWON Files: Binary format

TN 03

ver 1.1

11/10/2004

For all eWON® types

## 1 IrcAll.bin

---

The *ircall.bin* file contains the binary values of all Tags defined in the eWON. This file is an image of the Tag memory of the eWON.

This file is present on every eWON type, but this file is empty if the eWON cannot record data (eWON1000, eWON500, eWON2001).

**CAUTION: the processor of the eWON uses the Big Endian format for memory access (Most significant byte first) like Motorola processor. Thus, this access method is also applicable for the *ircall.bin* file. In PC world (Intel processor), the access method is Little Endian ! → We need to reverse all bytes read (and words if necessary) from files in order to correctly interpret it in a PC program.**

### 1.1 Header structure

The file begins by a header structure:

VersionHi		VersionLo		unused		StructLen	
15	0	15	0	15	0	15	0

This structure contains 4 short integers (16 bits): the firmware version of the eWON (VersionHi and VersionLo), an unused data (Dummy) and the length of the data structure (StructLen).

Example of implementation in C:

---

```
// struct declaration
typedef struct
{
    unsigned short VersionHi;
    unsigned short VersionLo;
    unsigned short dummy;
    unsigned short StructLen;
}
HistoricalHeader_t;

// variable declaration
HistoricalHeader_t Header;

In = fopen(FileName, "rb");// open file stream
fread(&Header, 8, 1, In);// read struct
```

---

The *swab* function is a standard C function for adjacent bytes swapping.

If your eWON is firmware version 1.3, the Header will be 1 3 0 16.

For all eWON® types

## 1.2 Data structure

After the 8 bytes of the Header, the eWON data can be found. Each data is encoded in a 16 bytes structure defined as follows:

LogTime		MSec	IntraSecCounter
TagId (31 bits)	Init	Value	

In C, structure declaration is as follow:

```
typedef struct
{
    time_t          LogTime;
    unsigned short  Msec;
    unsigned short  IntraSecCounter;
    unsigned int    InitValue:1;
    unsigned int    TagId:31;
    float          Value;
}
```

**LogTime:** time\_t (32 bits)

Absolute time in seconds (since 1970) for the data point.

**MSec:** unsigned short (16 bits)

This is the number of MSec to add to LogTime in order to get the complete timestamp of the data point.

**IntraSecCounter:** unsigned short (16 bits)

This value is incremented for each point logged during the same second (incremented even if TagId is different).

**InitValue:** bitfield (1 bit)

TRUE if the point was logged due to a restart of the system => not incremental then.

**TagId:** bitfield (31 bits)

Unique Id of the Tag (TagId in Var\_1st.txt), never the same for 2 Tags, even if tag deleted.

**Value:** float (32 bits)

Actual value (logged at the beginning of the measured interval).

**CAUTION: As this structure has a 4 bytes data type (time\_t, bitfield and float), you need to swap words and bytes for these data types.**



# eWON Files: Binary format

TN 03

ver 1.1

11/10/2004

For all eWON® types

---

Example of implementation in C:

```
// variables declaration
HistoricalHeader_t Header;
HistoricalRecord_t HstBuffer[100000];
char Buffer[1600000];

//-----
int ReadFile(char *FileName)
{
    FILE *In;
    long Len;

    if ((In = fopen(FileName, "rb")) == NULL)
    {
        fprintf(stderr, "Cannot open input file.\n");
        return 1;
    }

    Len = FileSize(In)-8;
    Len /= sizeof(HistoricalRecord_t);
    HstLen = Len;

    fread(&Header, 8, 1, In);
    swab((char*)&Header, (char*)&Header, 8);

    fseek(In, 8L, SEEK_SET);

    fread(Buffer, sizeof(HistoricalRecord_t), Len, In);
    SwapBytesAndWords((char*)Buffer, (char*)HstBuffer,
sizeof(HistoricalRecord_t)*Len);

    fclose(In);

    return 0;
}

//-----
long FileSize(FILE *Stream)
{
    long CurPos, Length;

    CurPos = ftell(Stream);
    fseek(Stream, 0L, SEEK_END);
    Length = ftell(Stream);
    fseek(Stream, CurPos, SEEK_SET);
    return Length;
}

//-----
void SwapBytesAndWords(char *Source, char *Dest, unsigned long Length)
```



# eWON Files: Binary format

TN 03

ver 1.1

11/10/2004

For all eWON® types

---

```
{  
    unsigned short *WordSource, *WordDest;  
    unsigned short tmp;  
  
    WordSource = (unsigned short*) Source;  
}
```

---

## 2 Inst\_val.bin

---

The *inst\_val.bin* file contains the instant values in binary format from of all Tags that are defined in the eWON.

## 3 dump.ppp

---

The *dump.ppp* file logs the eWON PPP communications (incoming, outgoing, or both), depending on the configuration you set in the **System Setup/General/PPP Dump** page. The frames that are captured in the log file can be parsed by the Ethereal software that you can download at this location:

<http://www.ethereal.com/download.html>