MetaStock File Format

Gianluca Scacco

Rel. 1

MetaStock File Format

Gianluca Scacco

Rel. 1

1. Introduction

Metastock is used from many tools to save information about financial data (stock quotes).

Data are stored in binary files in an old format: Microsoft Binary Format (QBasic !).

2. Directory Format

Metastock marcket data are called *securities;* each security is stored in a folder.

The folder contains one EMASTER file, one MASTER file and up to 256 Fx.DAT files. If the security stored in the folder has more than 256 stocks, in the folder there is also one XMASTER file and many Fn.MWD files.

The security folder contains different files:

1 **EMASTER:** Index file (see section 3)

2 **MASTER:** Index file (see section 4)

3 **Fx.DAT:** Single stock data file (see section 5)

4 **XMASTER:** Extended index file (if the number of stocks are grater then 256) (see section 6)

5 **Fn.MWD:** Single stock data file (see section 7)

3. EMASTER

EMASTER is an index file that contains the information on each stock stored in the security folder. The main information stored in this file are: *Stock name, Stock Symbol, First Date, Last Date, File number (the x in Fx.DAT), Last dividend paind* (see table 1 on page 2).

The first record is an header; the first byte contains the record count.

Start Byte	End Byte	Length	Description	Type
0	1	2	34h 31h Version number	Int ?
2	2	1	the value of x in Fx.DAT: from 0 to 255 !	Byte
3	10	8	Unknown	
11	24	14	Stock symbol: ends with a byte 0	String
25	31	7	Unknown	
32	47	16	Stock name: ends with a byte 0	String
48	63	16	Unknown	
64	67	4	First date format YYMMDD	Float CVS
68	71	4	Unknown	
72	75	4	Last date format YYMMDD	Float CVS
76	125	50	Unknown	
126	129	4	First date long format YYYYMMDD	Float CVL
130	130	1	Unknown	
131	134	4	Last dividend paid	Float CVL
135	138	4	Last dividend adjustment rate	Float CVS
139	191	53	Unknown	

Table 1. EMASTER: record length 192 bytes

Start Byte	End Byte	Length	Description	Type
0	0	1	the value of x in Fx.DAT: from 0 to 255 !	Byte
1	6	6	Unknown	
7	22	16	Stock name: ends with a byte 0	String
23	24	2	Unknown	
25	28	4	First date format YYMMDD	Float CVS
29	32	4	Last date format YYMMDD	Float CVS
33	35	3	Unknown	
36	49	14	Stock symbol: ends with a byte 0	String
51	52	3	Unknown	

Table 2. MASTER: record length 52 bytes

Start Byte	End Byte	Length	Description	Type
0	3	4	date format YYMMDD	Float CVMS

4	7	4	Open	Float CVMS
8	11	4	High	Float CVMS
12	15	4	Low	Float CVMS
16	19	4	Close	Float CVMS
20	23	4	Volume	Float CVMS
24	27	4	Open interest	Float CVMS

Table 3. Fx.DAT: record length 28 bytes

4. MASTER

The MASTER file duplicate the information contained in the EMASTER: don't know why ! (see table 2 on page 2)

The first record is an header.

5. Fx.DAT

Each file contains all the data of the stock.(see table 3 on page 3)

The first record is an header. The first 4 bytes contain the number of records.

The first record is an header.

6. XMASTER

The XMASTER file provides an index for the remaining Fn.MWD data files. It contains 150 bytes binary records: on header record and on record for each Fn.MWD file. (see table 4 on page 4)

The number contained are represented in integer format (litle endian byte order): shorts on two bytes and integer on four.

The header record contains the number of Fn.MWD files indexed: from byte 10 to byte 11 (short).

7. Fn.MWD

The format is the same of Fx.DAT. (see table 3 on page 3)

8. Conversion

The conversion:

CVS Function

Start Byte	End Byte	Length	Description	Type
0	0	1	Unknown	
1	15	15	Stock symbol: ends with a byte 0	String
16	61	46	Stock name: ends with a byte 0	String
62	62	1	’D’ maybe update type	Char
65	66	2	the number n in Fn.MWD	Short
67	79	13	Unknown	
80	83	4	End Date e.g. 19981125	Integer
84	103	20	Unknown	
104	107	4	Start Date	Integer
108	111	4	Start Date	Integer
112	115	4	Unknown	
116	119	4	End Date	Integer
120	149	30	Unknown	

Table 4. XMASTER: record length 150 bytes

```
’Convert from String to Single.
    Shared Function CVS(ByRef Argument As String) As Single
        Dim sTemp As Single = 0.0F
        If Len(Argument) <> 4 Then
            Return Single.NaN
        End If
        CopyMemoryCVS(sTemp, Argument, 4I)
        Return sTemp
    End Function
```

CVD Function

```
’Convert from String to Double.
    Shared Function CVD(ByRef Argument As String) As Double
        Dim dTemp As Double = 0.0R
        If Len(Argument) <> 8 Then
            Return Double.NaN
        End If
        CopyMemoryCVD(dTemp, Argument, 8I)
        Return dTemp
    End Function
```

CVL Function

```
’Convert from String to (QB)Long.
    ’QB/VB Long (4 bytes) => .NET Integer (Int32)
    Shared Function CVL(ByRef Argument As String) As Long
        Dim lTemp As Integer = 0I
        If Len(Argument) <> 4 Then
            Return Long.MinValue
        End If
        CopyMemoryCVL(lTemp, Argument, 4I)
        Return CLng(lTemp)  ’Cast Integer into Long
    End Function
```

CVI Function

```
’Convert from String to (QB)Integer.
    ’QB/VB Integer (2 bytes) => .NET Short (Int16)
    Shared Function CVI(ByRef Argument As String) As Integer
        Dim iTemp As Short = 0S
        If Len(Argument) <> 2 Then
            Return Integer.MinValue
        End If
        CopyMemoryCVI(iTemp, Argument, 2I)
        Return CInt(iTemp)  ’Cast Short into Integer
    End Function
```

CVSMBF Function

```
Shared Function CVSMBF(ByVal Num As String) As Single
        Dim Expon As Integer
        Dim Mant As Integer
        Dim NSign As Integer
        Dim Result As Single
        Result = 0
        If Len(Num) = 4 Then
            Expon = Asc(Right$(Num, 1)) - 128
            Mant = Asc(Mid$(Num, 3, 1))
            NSign = Mant \ 128
            Mant = 128 + Mant Mod 128
            Result = Mant / 256 + Asc(Mid$(Num, 2, 1)) / 256 ^ 2 _
                     + Asc(Left$(Num, 1)) / 256 ^ 3
            Result = Result * 2 ^ Expon
            If NSign Then
                Result = -Result
            End If
        End If
        Return Result
    End Function
```

CVDMBF Function

```
Public Function CVDMBF(ByVal Num As String) As Double
        Dim Expon As Integer
        Dim Mant As Integer
        Dim NSign As Integer
        Dim Cnt As Integer
        Dim Result As Double
        Result = 0
        If Len(Num) = 8 Then
            Expon = Asc(Right$(Num, 1)) - 128
            Mant = Asc(Mid$(Num, 7, 1))
            NSign = Mant \ 128
            Mant = 128 + Mant Mod 128
            Result = Mant / 256
            For Cnt = 6 To 1 Step -1
             Result = Result + Asc(Mid$(Num, Cnt, 1)) / 256 ^ (8 - Cnt)
            Next Cnt
            Result = Result * 2 ^ Expon
            If NSign Then
                Result = -Result
            End If
        End If
        CVDMBF = Result
    End Function
```

Fonctions on kernel32

```
Private Declare Sub CopyMemoryMKD Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByVal hDest As String, ByRef hSource As Double, _
        ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryCVD Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByRef hDest As Double, ByVal hSource As String, _
        ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryMKS Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByVal hDest As String, ByRef hSource As Single, _
        ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryCVS Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByRef hDest As Single, ByVal hSource As String, _
        ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryMKL Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByVal hDest As String, ByRef hSource As Integer, _
        ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryCVL Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByRef hDest As Integer, ByVal hSource As String, _
        ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryMKI Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByVal hDest As String, ByRef hSource As Short, ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryCVI Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByRef hDest As Short, ByVal hSource As String, ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryMKDt Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByVal hDest As String, ByRef hSource As Double, _
        ByVal iBytes As Integer)
    Private Declare Sub CopyMemoryCVDt Lib "Kernel32" Alias "RtlMoveMemory" _
        (ByRef hDest As Double, ByVal hSource As String, _
        ByVal iBytes As Integer)
```