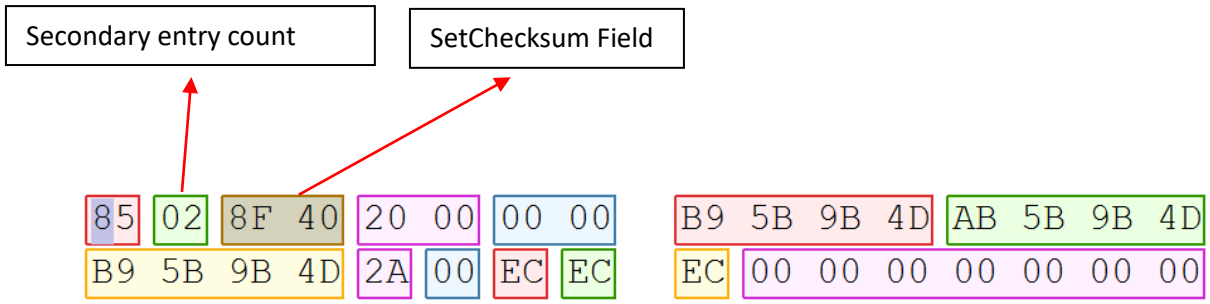
File Directory entry:

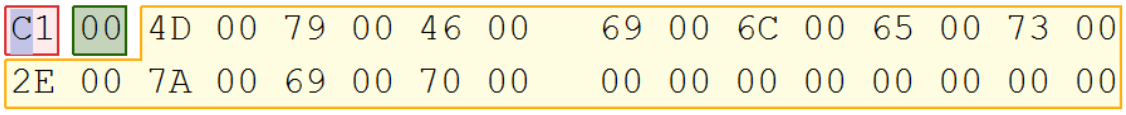


The SetChecksum field contains the checksum of all directory entries in the given directory entry set (https://docs.microsoft.com/en-us/windows/win32/fileio/exfat-specification section 6.3.3). In other words it is the checksum of all the File directory entry bytes (minus the two bytes used for itself).

Let's analyze how it works for the file entry above:
The entry has 2 secondary fields (as seen from the screenshot above), each 32 bytes long:

Entry 1:

Entry 2:



The checksum is calculated as follows:

NumberOfBytes = (Secondary Entry Count + 1 for the main File Directory entry) * 32.
In this example NumberOfBytes is 96 (32*3)

Starting with an initial Checksum value equal to zero, the following iteration adds a value calculated for each byte to the Checksum:

(If the current checksum value is an odd number it adds 32768 (0x8000)) + (Binary right shit of 1 of the current checksum value) + (the current byte's integer value).

Note: Bytes 2 and 3 of the first entry, namely the 'SetChecksum' field bytes are excluded.

#	byte	byte value	Checksum is odd	Checksum value	Checksum Binary value	Checksum Binary right shift 1 Decimal	Checksum Binary right shift 1	New Checksum	New Checksum (hex)
0	0x85	133		0	0000000000000000	0	0000000000000000	133	0x0085
1	0x02	2	Add 32768	133	0000000010000101	66	0000000001000010	32.836	0x8044
4	0x20	32		32.836	1000000001000100	16.418	0100000000100010	16.450	0x4042
5	0x00	0		16.450	0100000001000010	8.225	0010000000100001	8.225	0x2021
6	0x00	0	Add 32768	8.225	0010000000100001	4.112	0001000000010000	36.880	0x9010
7	0x00	0		36.880	1001000000010000	18.440	0100100000001000	18.440	0x4808
8	0xB9	185		18.440	0100100000001000	9.220	0010010000000100	9.405	0x24BD
9	0x5B	91	Add 32768	9.405	0010010010111101	4.702	0001001001011110	37.561	0x92B9
10	0x9B	155	Add 32768	37.561	1001001010111001	18.780	0100100101011100	51.703	0xC9F7
11	0x4D	77	Add 32768	51.703	1100100111110111	25.851	0110010011111011	58.696	0xE548
12	0xAB	171		58.696	1110010101001000	29.348	0111001010100100	29.519	0x734F
13	0x5B	91	Add 32768	29.519	0111001101001111	14.759	0011100110100111	47.618	0xBA02
14	0x9B	155		47.618	1011101000000010	23.809	0101110100000001	23.964	0x5D9C
15	0x4D	77		23.964	0101110110011100	11.982	0010111011001110	12.059	0x2F1B
16	0xB9	185	Add 32768	12.059	0010111100011011	6.029	0001011110001101	38.982	0x9846
17	0x5B	91		38.982	1001100001000110	19.491	0100110000100011	19.582	0x4C7E
18	0x9B	155		19.582	0100110001111110	9.791	0010011000111111	9.946	0x26DA
19	0x4D	77		9.946	0010011011011010	4.973	0001001101101101	5.050	0x13BA
20	0x2A	42		5.050	0001001110111010	2.525	0000100111011101	2.567	0x0A07
21	0x00	0	Add 32768	2.567	0000101000000111	1.283	0000010100000011	34.051	0x8503
22	0xEC	236	Add 32768	34.051	1000010100000011	17.025	0100001010000001	50.029	0xC36D
23	0xEC	236	Add 32768	50.029	1100001101101101	25.014	0110000110110110	58.018	0xE2A2
24	0xEC	236		58.018	1110001010100010	29.009	0111000101010001	29.245	0x723D
25	0x00	0	Add 32768	29.245	0111001000111101	14.622	0011100100011110	47.390	0xB91E
26	0x00	0		47.390	1011100100011110	23.695	0101110010001111	23.695	0x5C8F
27	0x00	0	Add 32768	23.695	0101110010001111	11.847	0010111001000111	44.615	0xAE47
28	0x00	0	Add 32768	44.615	1010111001000111	22.307	0101011100100011	55.075	0xD723
29	0x00	0	Add 32768	55.075	1101011100100011	27.537	0110101110010001	60.305	0xEB91
30	0x00	0	Add 32768	60.305	1110101110010001	30.152	0111010111001000	62.920	0xF5C8
31	0x00	0		62.920	1111010111001000	31.460	0111101011100100	31.460	0x7AE4
32	0xC0	192		31.460	0111101011100100	15.730	0011110101110010	15.922	0x3E32

#	byte	byte value	Checksum is odd	Checksum value	Checksum Binary value	Checksum Binary right shift 1 Decimal	Checksum Binary right shift 1	New Checksum	New Checksum (hex)
33	0x03	3		15.922	0011111000110010	7.961	0001111100011001	7.964	0x1F1C
34	0x00	0		7.964	0001111100011100	3.982	0000111110001110	3.982	0x0F8E
35	0x0B	11		3.982	0000111110001110	1.991	0000011111000111	2.002	0x07D2
36	0x77	119		2.002	0000011111010010	1.001	0000001111101001	1.120	0x0460
37	0x09	9		1.120	0000010001100000	560	0000001000110000	569	0x0239
38	0x00	0	Add 32768	569	0000001000111001	284	0000000100011100	33.052	0x811C
39	0x00	0		33.052	1000000100011100	16.526	0100000010001110	16.526	0x408E
40	0x32	50		16.526	0100000010001110	8.263	0010000001000111	8.313	0x2079
41	0x5F	95	Add 32768	8.313	0010000001111001	4.156	0001000000111100	37.019	0x909B
42	0xA2	162	Add 32768	37.019	1001000010011011	18.509	0100100001001101	51.439	0xC8EF
43	0x00	0	Add 32768	51.439	1100100011101111	25.719	0110010001110111	58.487	0xE477
44	0x00	0	Add 32768	58.487	1110010001110111	29.243	0111001000111011	62.011	0xF23B
45	0x00	0	Add 32768	62.011	1111001000111011	31.005	0111100100011101	63.773	0xF91D
46	0x00	0	Add 32768	63.773	1111100100011101	31.886	0111110010001110	64.654	0xFC8E
47	0x00	0		64.654	1111110010001110	32.327	0111111001000111	32.327	0x7E47
48	0x00	0	Add 32768	32.327	0111111001000111	16.163	0011111100100011	48.931	0xBF23
49	0x00	0	Add 32768	48.931	1011111100100011	24.465	0101111110010001	57.233	0xDF91
50	0x00	0	Add 32768	57.233	1101111110010001	28.616	0110111111001000	61.384	0xEFC8
51	0x00	0		61.384	1110111111001000	30.692	0111011111100100	30.692	0x77E4
52	0x01	1		30.692	0111011111100100	15.346	0011101111110010	15.347	0x3BF3
53	0x02	2	Add 32768	15.347	0011101111110011	7.673	0001110111111001	40.443	0x9DFB
54	0x00	0	Add 32768	40.443	1001110111111011	20.221	0100111011111101	52.989	0xCEFD
55	0x00	0	Add 32768	52.989	1100111011111101	26.494	0110011101111110	59.262	0xE77E
56	0x32	50		59.262	1110011101111110	29.631	0111001110111111	29.681	0x73F1
57	0x5F	95	Add 32768	29.681	0111001111110001	14.840	0011100111111000	47.703	0xBA57
58	0xA2	162	Add 32768	47.703	1011101001010111	23.851	0101110100101011	56.781	0xDDCD
59	0x00	0	Add 32768	56.781	1101110111001101	28.390	0110111011100110	61.158	0xEEE6
60	0x00	0		61.158	1110111011100110	30.579	0111011101110011	30.579	0x7773
61	0x00	0	Add 32768	30.579	0111011101110011	15.289	0011101110111001	48.057	0xBBB9
62	0x00	0	Add 32768	48.057	1011101110111001	24.028	0101110111011100	56.796	0xDDDC
63	0x00	0		56.796	1101110111011100	28.398	0110111011101110	28.398	0x6EEE

#	byte	byte value	Checksum is odd	Checksum value	Checksum Binary value	Checksum Binary right shift 1 Decimal	Checksum Binary right shift 1	New Checksum	New Checksum (hex)
64	0xC1	193		28.398	0110111011101110	14.199	0011011101110111	14.392	0x3838
65	0x00	0		14.392	0011100000111000	7.196	0001110000011100	7.196	0x1C1C
66	0x4D	77		7.196	0001110000011100	3.598	0000111000001110	3.675	0x0E5B
67	0x00	0	Add 32768	3.675	0000111001011011	1.837	0000011100101101	34.605	0x872D
68	0x79	121	Add 32768	34.605	1000011100101101	17.302	0100001110010110	50.191	0xC40F
69	0x00	0	Add 32768	50.191	1100010000001111	25.095	0110001000000111	57.863	0xE207
70	0x46	70	Add 32768	57.863	1110001000000111	28.931	0111000100000011	61.769	0xF149
71	0x00	0	Add 32768	61.769	1111000101001001	30.884	0111100010100100	63.652	0xF8A4
72	0x69	105		63.652	1111100010100100	31.826	0111110001010010	31.931	0x7CBB
73	0x00	0	Add 32768	31.931	0111110010111011	15.965	0011111001011101	48.733	0xBE5D
74	0x6C	108	Add 32768	48.733	1011111001011101	24.366	0101111100101110	57.242	0xDF9A
75	0x00	0		57.242	1101111110011010	28.621	0110111111001101	28.621	0x6FCD
76	0x65	101	Add 32768	28.621	0110111111001101	14.310	0011011111100110	47.179	0xB84B
77	0x00	0	Add 32768	47.179	1011100001001011	23.589	0101110000100101	56.357	0xDC25
78	0x73	115	Add 32768	56.357	1101110000100101	28.178	0110111000010010	61.061	0xEE85
79	0x00	0	Add 32768	61.061	1110111010000101	30.530	0111011101000010	63.298	0xF742
80	0x2E	46		63.298	1111011101000010	31.649	0111101110100001	31.695	0x7BCF
81	0x00	0	Add 32768	31.695	0111101111001111	15.847	0011110111100111	48.615	0xBDE7
82	0x7A	122	Add 32768	48.615	1011110111100111	24.307	0101111011110011	57.197	0xDF6D
83	0x00	0	Add 32768	57.197	1101111101101101	28.598	0110111110110110	61.366	0xEFB6
84	0x69	105		61.366	1110111110110110	30.683	0111011111011011	30.788	0x7844
85	0x00	0		30.788	0111100001000100	15.394	0011110000100010	15.394	0x3C22
86	0x70	112		15.394	0011110000100010	7.697	0001111000010001	7.809	0x1E81
87	0x00	0	Add 32768	7.809	0001111010000001	3.904	0000111101000000	36.672	0x8F40
88	0x00	0		36.672	1000111101000000	18.336	0100011110100000	18.336	0x47A0
89	0x00	0		18.336	0100011110100000	9.168	0010001111010000	9.168	0x23D0
90	0x00	0		9.168	0010001111010000	4.584	0001000111101000	4.584	0x11E8
91	0x00	0		4.584	0001000111101000	2.292	0000100011110100	2.292	0x08F4
92	0x00	0		2.292	0000100011110100	1.146	0000010001111010	1.146	0x047A
93	0x00	0		1.146	0000010001111010	573	0000001000111101	573	0x023D
94	0x00	0	Add 32768	573	0000001000111101	286	0000000100011110	33.054	0x811E
95	0x00	0		33.054	1000000100011110	16.527	0100000010001111	16.527	0x408F

The resulting Checksum value after the iteration is 16527 or 0x408F which matches the little-endian value of 0x8F40 seen in the File Directory entry ‘SetChecksum’ field.

Sample powershell script to calculate the Checksum value:

```
# Reference: https://docs.microsoft.com/en-us/windows/win32/fileio/exfat-specification
# chapter "6.3.3 SetChecksum Field"

# Clear screen
Clear-Host

# get bytes from the user
$bytes = Read-Host -Prompt "Input File directory entry bytes (eg'85028F4020000000B95B9')"
# Split bytes to pairs
$bytes = $bytes -split '(..)'| ? { $_ }
#Count number of bytes
$NumberOfBytes = $bytes.count

# Set initial checksum value to 0
$Checksum = 0

# Start the iteration
for ($Index = 0; $Index -lt $NumberOfBytes; $Index++){

# Exclude 'SetChecksum' field bytes from the calculation
if($index -notin (2,3)){

# Get the decimal value of the current hex byte
$int = [int]"0x$($bytes[$index].PadLeft(2,'0'))"

# check if current value of the checksum is odd number
If ($Checksum -band 1) {$h = 0x8000} Else {$h = 0}

# Calculate checksum for current byte
$Checksum = $h + ($Checksum -shr 1) + $int
} # if loop end

} # Iteration ends

# Convert Checksum to little Endian to match the ExFat directory entry stored value
# for easier verification.

$Check = "{0:X}" -f $Checksum
# Split each byte hex value
$Check = $Check -split '(..)'
# Reverse byte order
[array]::reverse($Check)
# Join the reversed bytes together
$Check = -join $Check

# Write resulting Checksum to console
Write-Host "The Checksum value is: " -f White -nonewline;Write-Host $Check -f Yellow
```