There should be two formats. One for the book ala polyglot, and one for the persistent transposition table. It's not really possible to do both in one, as TT requires additional data and is more engine specific. However it should still be possible to create an initial book from data in the transposition table (but it will be one way only).

All integer values are little endian

Book

Header (32 bytes)

"YACB" : (4 bytes)
    magic. "Yet Another Chess Book"
version : (1 byte)
    must be 0
reserved : (27 bytes)
    must be 0

Zobrist (781 * 8 = 6248) bytes

    present iff key_type == 1
    order is polyglot order http://hgm.nubati.net/book_format.html

Entry (16 bytes)

key : Stockfish 64 bit Zobrist (8 bytes)
    ~6 billion positions to get >=1 expected number of collisions
    keys in the file must be unique and in ascending ordered
    this allows just using the TT hash key for the initial creation
this can be arbitrary, for example log2(nodes)
elo : (2 bytes)
    the estimated elo of the entry, that is the estimated elo rating the engine would achieve if it played using similar analysis as what went into this entry against the baseline
        the baseline is Stockfish 14 with 1 thread at 1M nodes per move. That is an entry evaluated with Stockfish 14 with 1 thread at 1M nodes would have elo = 0
        this requires some empirical function of node count and thread count
    used for entry replacement. Can be set to 2**16-1 to make an entry irreplaceable
value_meta : (1 byte)
    value_type : (3 highest bits)
        0 - centipawns for white
        1 - centipawns for white after known win threshold
        2 - DTZ for white
        3 - DTM for white
        4 - perf% for white (0 = 0%, 40000 = 100%, 400 = 1%, 1 = 0.0025%, values outside of 0..40000 not allowed)
        5..7 - reserved
    bound_type : (next 2 highest bits)

0 - exact
1 - lower bound from white perspective
2 - upper bound from white perspective
3 - proven (for example a proven DTZ or DTM value from TB, sufficiently confident search resulting in a proven value. Other values should not be marked proven, use exact)

source : (next 2 highest bits)
0 - TT
1 - engine analysis
2 - statistical
3 - other

reserved : (lowest 1 bit)

reserved : (1 byte)
must be zero, no idea what to put in there but we should pad to 16 bytes

value : (2 bytes)

best_move : Stockfish Move (2 bytes)
the engine must check if this move is legal. it might not be due to a hash collision

Persistent TT

Just for stockfish. All >1 byte values are stored in little-endian byte order

Header (128 bytes)

“SFTT” : uint8_t[4]

version : uint8_t

root_pos : uint8_t[91]
last analysed fen, padded with ‘\0’
or completely \0 if not saved

nodes : uint64_t
number of nodes searched
or 0 if not searched

root_depth : uint8_t
the last finished root depth
or 0 if not searched

min_entry_depth : uint8_t
we don’t want to save all the entries, size reduction is important

reserved : uint8_t[22]
must be 0

Entry (16 bytes)

key64 : uint64_t

depth8 : uint8_t

gen_bound8 : uint8_t

move : uint16_t

value : uint16_t
eval : uint16_t