SAR Trigger and Action Table Format

SAR Trigger Tables and SAR Action Tables are each stored as separate UEFI data structures. Each structure stores an array of Trigger or Action tables, so from the UEFI perspective there is exactly one SAR Trigger Table structure holding all Trigger Tables, and one SAR Action Table structure holding all Action tables.

Overall layout for one UEFI Trigger or Action table structure:

|  |  |  |
| --- | --- | --- |
| Name | Data Type/size | Description |
| TableCount | uint32\_t | Number of SAR Trigger/Action tables. |
| Offset Array | uint32\_t \* TableCount | Array of offset values to following embedded SAR Tables.  The total number of values in the array equals TableCount.  Each offset holds a byte count of the offset to an embedded SAR table. An offset of 0 represents a table starting immediately after this list of offset values, so the offset to the first SAR table will typically be 0.  Note that it is legal for multiple tables to share the same underlying data by reusing the same offset. So, for example, on a system with four identical SAR trigger tables, this could be represented as a TableCount of 4, followed by four offset values each set to 0. |
| SAR Tables Array | Variable Size | One or more SAR trigger or action tables. See description below. |

SAR Tables structure

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| NumCols | uint8\_t | Number of columns in the table. Note that the maximum is 255, if we anticipate more then we should increase to uint16\_t. |
| NumRows | uint8\_t | Number of rows in the table. Note that the maximum is 255, if we anticipate more then we should increase to uint16\_t. |
| SAR Table Metadata  Array | Variable Size | An array SAR Table Metadata Entries. The number of entries in the array is equal to the number of columns. Format of each entry is described below. |
| SAR Table Data  Array | Variable Size | Data representing triggers or actions driving SARManager behavior. The number of entries in this array is equal to the number of columns multiplied by the number of rows. The size of each entry is determined by the ByteCount field of the corresponding SAR Table Metadata Entry for the column in which the entry is located. |

SAR Table Metadata Entry

|  |  |  |
| --- | --- | --- |
| Name | Data Type | Description |
| Type | uint8\_t | Enum value representing the type of the data.  For a trigger table, this might be a value representing testing Posture, PES Sensor, Motion, etc.  For an action table, this might be a value representing setting LTE Power or WiFi Power.  Note that uint8\_t limits the total number of trigger or action types to 255. |
| ByteCount | uint8\_t | Number of bytes required for data values within this column. A simple trigger that just needs to support three states of triggered, not triggered, or don’t care would just require one byte of data. On the other hand, a more complex trigger, such as matching against a specific hinge angle, might require four bytes of data representing two 16-bit values of min and max hinge angle. Likewise, an action for setting WiFi SAR backoff might require four bytes to control Chain A enable, Chain B enable, Chain A power, and Chain B power.  Note that uint8\_t limits the size of data associated with a trigger or action to 255 bytes. |

SAR Table Data Array

|  |  |  |
| --- | --- | --- |
| Row 0/Col 0 data | Variable | Size specified by metadata byte count for column 0. |
| Row 0/Col 1 data | Variable | Size specified by metadata byte count for column 1. |
| … |  |  |
| Row 0/Col N data | Variable | Last data entry in the first row, where N =NumCols-1 |
| Row 1/Col 0 data | Variable |  |
| Row 1/Col 1 data | Variable |  |
| … |  |  |
| Row M/Col N | Variable | Last data entry in the table, where M=NumRows-1, N=NumCols-1 |

Example: The following is a trivial example of a set of four SAR trigger tables, where:

* All four trigger tables share the same trigger table offset and therefore data.
* There are two triggers defined with the following Enum type values:
  + PES = 0
  + Simultaneous Transmit = 1
* Each trigger requires one byte of data, e.g. to represent matching state on or off

static const uint8\_t TriggerTableData[] =

{

4,0,0,0, // TableCount: 4 tables (little endian)

0,0,0,0, // Table 0 offset

0,0,0,0, // Table 1 offset

0,0,0,0, // Table 2 offset

0,0,0,0, // Table 3 offset

// SAR Table (the only table in this example)

3, 4, // NumCols, NumRows

// Metadata: defining types and sizes for SAR table data

0, 1, // Pes needs one byte of data (on/off)

1, 1, // Simultaneous transmit needs one byte of data (on/off)

// Data: Defining actual triggers or actions.

// Pes SimultaneousXmit

0, 0, // Matches the case where PES=0, SimultaneousXmit=0

0, 1, // Matches the case where PES=0, SimultaneousXmit=1

1, 0, // Matches the case where PES=1, SimultaneousXmit=0

1, 1 // Matches the case where PES=1, SimultaneousXmit=1

// If there were additional SAR tables, the would start here, and

// have a nonzero offset in the offset array.

};