Managing User Memory

Some Gen 2 RFID tags have 96 bits of EPC memory, just enough to contain an Electronic Product Code that serves as a unique identifier of the object to which the tag is affixed. Any additional information about the tagged object can be associated with the EPC in an external database. Other tags have a user memory bank of 96, 128, 256 or more bits of memory, so data can be stored on the tag.

A tag on a package of perishable goods, for instance, could carry the product expiration date and batch number, and a tag on a shipping container could include the mailing address of the recipient. A tag on an aircraft part or oil pipeline valve could store the asset's maintenance history; each time the asset is inspected, a new piece of historic data is added to the tag's user memory.

But setting up user memory so you can effectively read and write data to a tag is challenging, so it's important to work with a software vendor or systems integrator that has experience managing the following issues.

User memory is stored in 16-bit blocks, and you must decide what data to store in each block. Then, you need to configure the software that controls your reader—either middleware or an application embedded in the reader—so it knows which data to access depending on the application. How you "lay out" your data in user memory is critical because layout can affect performance. A reader must send commands to the tag to access user memory, a process that takes time and can lower read rates.

Because user memory has limited storage capacity, it's also necessary to compress the data. The widely accepted ISO 15962 standard provides several compression methods, but each has trade-offs—among them, space versus performance. If an application must read all data elements simultaneously, you can compress the elements together, minimizing the number of bits required for storage. But the more data elements that must be read together, the slower the read rate. If an application needs to read only a few data elements at a time,

a better option is to compress the elements in separate user memory blocks. This approach requires fewer commands, so it yields better performance, though it uses more memory overall.

In a closed-loop application, you can use a proprietary format to encode data in user memory. But if you need to share the data with your business partners, you'll have to use one of the standards designed for supply-chain applications: GS1's Application Identifiers, for consumer supply chains; ANSI Data Identifiers, for the manufacturing industry; or Text Element Identifiers, for aerospace. Each provides dozens of commonly used descriptive data elements, such as expiration date, batch number, dimensions and weight, and defines standardized codes that identify each data element in user memory.

Ken Traub is the founder of Ken Traub Consulting, a Mass.-based firm providing services to software product companies and enterprises that rely on advanced software technology to run their businesses. Send your software questions to swsavvy@kentraub.com.