

that is needed (see **FIG. 3**). This sort of fixed-size executable image becomes a re-usable software/firmware part.

**[0109]** **FIG. 3** depicts a deployment architecture according to embodiments of the present invention. This diagram depicts an application of this invention to an embedded firmware part. This particular firmware part features a fixed-size firmware image divided into three (potentially fixed-sized) parts: boot block, description block and operational block.

**[0110]** As shown in **FIG. 3**, a deployment architecture according to embodiments of the present invention includes a configuration utility, a host and an optional description repository. The configuration utility is used to describe host ASICs and peripheral devices (e.g., for some entire product line), to configure a part after product assembly. (Note that the part is configured, not customized). The host (ASIC, etc.) includes a description block and a boot (image) block. The description block contains logical a physical descriptions and describes both the host ASIC and peripheral devices. It can pre-allocate a fixed amount of storage (to form a so-called, fixed-image firmware/software “part”) or it can dynamically allocate storage from a pool (which may or may not include off-chip/secondary storage). The host declares schemata types known (self-describing). The boot block is a fixed image which calls described/parameterized initializers (especially for each peripheral device present) and mediates and installs firmware and software updates, as needed. An operational block is a fixed image parameterized via descriptions.

**[0111]** A dynamic discovery monitor (DDM) can be used to enable collaboration with a centralized description repository. In some embodiments of the present invention, the DDM continuously detects peripherals, activates or uploads descriptions and/or monitors peripheral presence.

**[0112]** An optional description repository describes the universe of (known/supported) host ASICs; the universe of (known/supported) peripheral devices and then supplies needed (physical/logical) descriptions on-demand (based, e.g., on discovery of new devices by the DDM).

Host (ASIC) Descriptions Complement Peripheral Descriptions

**[0113]** A collection/container/table of logical descriptions is used when describing a host/ASIC processing support platform (for some software/firmware part). Each description (block entry) corresponds to an instance of some corresponding host ASICs circuit/sub-system. A similar collection/container/table of logical descriptions usually exists for each peripheral device. Together, these logical (and corresponding descriptions) comprise the description block. (see **FIG. 3**).

**[0114]** Generally, the distinction between what is an ASIC circuit/component-system and what is a peripheral device/component-system is not always precise. This makes it quite natural to unify the nature of these descriptions. This invention applies equally well to both host and peripheral support needs.

**[0115]** Again, the logical and physical host descriptions are conceptually distinct. The logical host aspect/component descriptions supply values that eventually map to some named/identified physical description. As with peripheral

descriptions, any one-to-one logical-to-physical identity mappings make it possible to interleave the two descriptions. As before, this is typically done as an implementation convenience. Prototype schemata may be created for ASIC circuits/component-systems like a random number generator, a UART port, non-volatile storage, tachometer circuits, pulse width modulator circuits and the like. A collection of such prototype schemes comprises a host schema which, when combined with a peripheral schema, form a (master) system schema.

**[0116]** Note that host (ASIC) and peripheral (device) descriptions can be mix-and-matched. For example, a palette of individual peripheral schemata can define a peripheral schema for some (particular vendors) product line. As new models enter this product line, they might regularly re-use the same peripheral schema. They might or might not re-use the same host schema. The host schema can be swapped in/out quite independently of the peripheral schema.

**[0117]** Those skilled in the art will immediately realize that many other sorts of useful scheme groupings can be affected. Other than for size constraints, a fixed-size executable image, according to embodiments of the present invention, may support all (or a very large number) of individual prototype schemes (for both various hosts and various peripherals). As a practical (and economic matter), carrying around unused schemes is hard to justify (past a certain degree). Furthermore, the executable image itself will have host instruction set dependencies (i.e., it will depend on the micro-controller processing core—e.g., H8, ARM, etc.).

**[0118]** This invention provides for a system wherein there is rarely any requirement for any embedded firmware/software part changes (especially across a select product line). The host hardware can be designed, and then desired software/firmware part(s) can be layered on. Nothing needs to be re-compiled, nothing needs to be re-initialized, and no run-time changes to the embedded firmware/software image are required (no overlay changes, no dynamic linking, no dynamic loading, etc.). A single, fixed software/firmware image can be used across a range of host hardware and across a range of peripheral devices. The product line designer matches the range of applicability of his embedded firmware/software part to his product line plans. This dramatically speeds up integration of the part and dramatically reduces time-to-market for new product model development, while dramatically reducing quality assurance costs. In addition, this makes it possible to support many/most hardware changes in the field (through small descriptive declarations). In fact, adaptation to hardware changes can be fully automated (via a dynamic discovery monitor—see **FIG. 3**).

Dynamic Hardware Environments

**[0119]** In practice, a host hardware environment rarely changes dynamically (at run-time). Once a product is shipped, the host hardware environment is typically fixed. If the software/firmware host is an ASIC (chip), it might even be fixedly connected to a printed circuit board (PCB). Although rare, an ASIC can change. For example, complementary support chips might be populated into waiting sockets. The host ASIC chip itself can be in a socket. This means that it too might be subject to field upgrades. Alternatively, the host might be on a card (i.e., a daughter card, mezzanine card, bus card, etc.). These too can be dynamically added, removed, swapped, etc.