Plug-and-Play Specification Introduced

Microsoft, Intel, Others Announce User-Friendly ISA Bus

By Dean McCarron

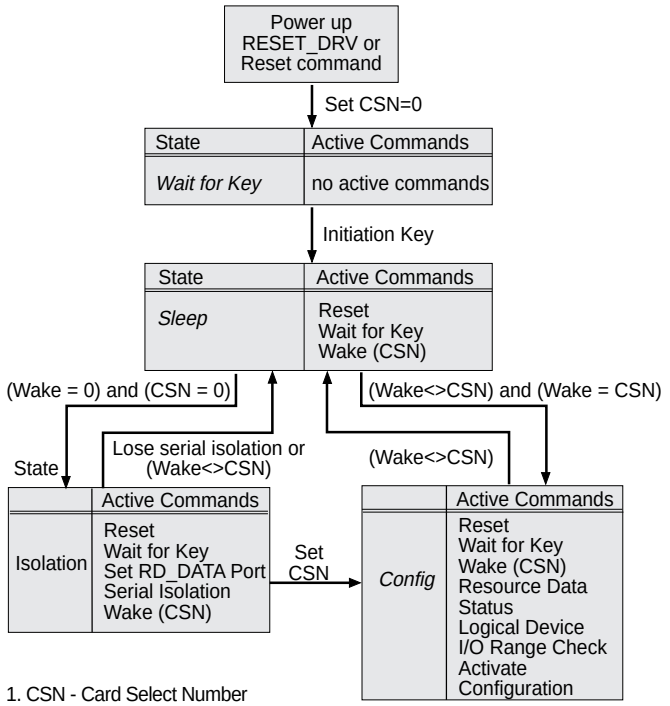
Dean McCarron is a senior analyst in our new MicroSystems Group. He recently joined us after six years at In-Stat, a semiconductor analysis firm. This is his first article for Microprocessor Report.

The emerging Plug-and-Play ISA specification spearheaded by Microsoft, Intel, and 11 others is shaping up to deliver on the promise of its name—at least for new systems. For the installed base of some 80 million AT- and PC-class machines, however, Plug and Play appears much like a dog trained to play chess: it doesn't play a very good game, but that it can play at all is impressive.

The goal of the Plug and Play (PNP) initiative, unveiled at Microsoft's Windows Hardware Engineering Conference in March, is to make PCs easier to use. The PNP ISA specification supports this goal. It would make adding a peripheral card to a PC more like an Apple Macintosh by eliminating tinkering with DIP switches, jumpers, IRQ settings, and I/O ports. PNP promises auto-configurable PCs by identifying and configuring add-in cards, and then loading the appropriate drivers, without user intervention.

Due to PNP's retrofit approach to the ISA bus, the operation needs a detailed explanation—remember that PNP's intent is to work in ISA-based PCs, old and new, and hence is required to handle existing, non-PNP cards and function in a sometimes hostile environment.

The details of PNP operation below are based on the recently-released version 0.9 of the specification. Microsoft plans to release version 1.0 later this month. Most of the procedures described here are expected to remain stable.



1. CSN - Card Select Number
2. RESET_DRV or the Reset Command causes a state transition from the current state to Wait for Key and sets all CNSs to zero.
3. The Wait for Key command causes a state transition from the current state to Wait for Key.

Figure 1. The state transitions of a Plug-and-Play ISA card.

PNP Operation

The sequence of events for PNP operation is straightforward; the execution of the events is somewhat more complicated. It operates in four distinct states, which are selected by the configuration software running on the PC. The four states of a PNP card are:

1. Wait For Key—All PNP cards wait for the initiation key before accepting any other commands. Once a valid key is received, a card shifts to the Sleep state.
2. Sleep—In this state, a card watches for PNP commands on the ISA bus but otherwise remains inactive, keeping its data bus drivers in high-impedance mode. A card not yet assigned a unique Card Select Number (CSN), has a CSN of zero. The card transitions into the Isolation or Config state when given a WAKE command, depending on the value of the CSN, as shown in Figure 1.
3. Isolation—A card in this state processes the isolation protocol, a complicated process that arbitrates among PNP cards in the system to create simulated geographic addressing. Cards that lose a given round of the protocol go back to the Sleep state. A single card wins the arbitration and has its CSN register programmed with a unique number. Once the CSN is set, the card moves into the Config state.
4. Config—In the Config state, a card's resource requirement data can be read and the base address, IRQs and DMA channels can be configured. The card can then be activated. During isolation or configuration activities involving other PNP cards, a card in this state will go back to Sleep.

The PNP configuration software—which can be in the BIOS, the device driver, or the operating system—drives the PNP cards through each state. The software first broadcasts an initiation key to all PNP cards, driving them to the Sleep state. The software then causes a

transition to the Isolation state and performs the isolation protocol, which ultimately allows each card to be uniquely identified with a card select number. Once all cards have been identified, each card is configured according to its hardware requirements.

Card Initialization

When an ISA PC is first turned on or after a hard reset, all PNP peripherals—both those on the motherboard and those on add-in cards—enter the Wait For Key state. The peripherals (except boot devices) are not allowed to drive the ISA bus and await an initiation key.

Boot devices are treated differently from non-boot PNP cards. Boot devices—peripherals that are integral to system startup—must remain active while other PNP devices are isolated from the ISA bus during a cold boot. In a basic PNP system, the hard and floppy disk controllers, keyboard controller, and graphics adapter are defined as boot devices. For diskless PCs, a network adapter is configured as a boot device in place of the disks.

PNP boot devices present a thorny problem. Since boot devices are active at startup, the potential exists for hardware conflicts between two or more active devices, PNP or not. As a result, boot devices may require what PNP is supposed to eliminate: tinkering with DIP switches and jumpers. That hurdle could be cleared later this year, when systems are expected to begin shipping with a PNP-aware BIOS. Boot devices in older PCs may still require intervention unless the BIOS is upgraded to a PNP-aware version.

Any configuration set by jumpers or other hardware for a PNP boot device can be overridden once the system has booted and the operating system's PNP drivers take over. This allows PNP to configure the system for optimum performance, or to avoid resource conflicts between a boot device and another PNP card that had been isolated during the system boot.

The system then boots the operating system from the designated boot device, which could be a disk controller or network card. At this point, PNP configuration software is loaded, either as independent drivers such as those in the current DOS environment, or as part of the operating environment itself, as promised in the "Chicago" version of Microsoft Windows currently under development. In either case, the steps that the software takes next are identical.

The purpose of the following process is to force all PNP cards into a known state in a way that is conflict-free for the ISA environment.

The PNP configuration software first performs a sequence of writes to the write-only PNP address port, which is I/O port address 0x0279 in a PC system. This address is the printer (LPT2) status port, which was chosen because in current systems it is a read-only port that is unaffected by writes.

The data written to the port is an initiation "key" that enables the PNP interface logic on all PNP devices in the system. The purpose of having an enabling key for PNP cards is to prevent ordinary I/O activity to ISA cards from inadvertently activating a PNP card. This key consists of a 32-byte sequence that is generated from a tapped linear feedback shift register (LFSR). A LFSR can be implemented in a card's control IC with fewer resources than a ROM would require. The interface logic on each PNP card compares the key sequence written to the PNP address port against an internally generated key based on an identical LFSR. Once there is a match between the two keys, all PNP cards go into the Sleep state, where they remain isolated from the data bus but will accept commands.

Commands are broadcast to all PNP cards by writing the address of the command register to the same address port as the initiation key. Data for the command is then written to the WRITE_DATA port, at the address of 0x0A79, which is an alias of the LPT2 status port. Once data is written to a command register, the command is executed. Only those cards which match the CSN argument of the last WAKE command with their own card CSN will execute a command, however.

Once the PNP cards are unlocked by the initiation key, the configuration software issues a WAKE(CSN) command, with a CSN of zero. Since all PNP cards default at reset to a zero CSN, all PNP cards respond to this WAKE command. When a card with a CSN of zero gets a WAKE command, it responds by entering the Isolation state; therefore, all PNP cards now enter the Isolation state.

The first time the Isolation state is entered, the configuration software will broadcast the SET READ_DATA command to all PNP cards to assign a data read port in the range of 0x0200 to 0x03FF in ISA I/O space. The read port address is variable to prevent any possible conflicts with existing non-PNP ISA cards. The second and subsequent times the Isolation state is entered, the read data port is not set, unless there has been a conflict. If any conflict—which can be detected by invalid data reads—occurs during the PNP configuration process, the software returns to the Isolation state and uses the SET READ_DATA command to set a new and hopefully conflict-free read address.

Serial Isolation Protocol

Once all the cards have been initialized, the PNP isolation process, which is a way of simulating geographic addressing, begins. The isolation process is highly repetitive. It consists of placing all PNP cards through the isolation protocol until only one card remains, assigning that card a new, non-zero card select number, and telling it to "sleep." Another WAKE command is then issued to cards with a zero CSN; previously isolated cards have a non-zero CSN and will not respond. This

process is repeated so eventually each card is isolated and given a non-zero CSN.

This isolation protocol is based on a 72-bit card ID, which is composed of a 32-bit manufacturer ID (identical to the manufacturer's EISA ID), a 32-bit serial number, and an 8-bit checksum. The serial number allows two or more identical cards to coexist in a system. Should a manufacture desire, the serial number can be set to all 1's, but this precludes more than one of this specific card per system.

To isolate a card, the software issues 72 repetitions of two reads from the read port address. For the first of the two reads, cards examine the current bit of their ID. Cards with a "1" bit in the current position drive a 0x55 on the data bus in response to the read, while cards with a "0" bit place their data bus drivers in high-impedance mode and watch for a 0x01 to appear on the bottom two bits of the data bus. Note that 0x01 is the bit pattern of the lower two bits when a 0x55 is driven onto the data bus; therefore, a PNP card with a "0" bit in its ID can detect if another card has a "1" bit.

The 0x01 bit pattern is not unique enough to reliably detect other cards in a PNP system; an ISA card or even a floating bus could cause this pattern to appear. To overcome this potential fault, the ID bit is read again by the second of the two reads. In response to the second read, however, a device that has a current bit of "1" drives 0xAA on the data bus, and cards with a "0" bit go into high-impedance mode and watch for 0x10 on the bottom two bits. A 0x10 bit pattern is the lower two bits when a 0xAA is driven onto the bus; because of the toggled bit patterns, a PNP card can detect other cards with high reliability. At this point, any card with a zero bit in the current ID position that saw a sequence of 0x01, 0x10 on the lower two data bits "drops out" of the isolation process for the current round and goes into the Sleep state.

The software then waits 250 microseconds before performing the next two-read sequence to allow the remaining PNP cards time to read the next bit of their serial ID, which is typically kept in an inexpensive but slow serial PROM. After the delay, another two-read sequence plus delay is repeated until all 72 bits have been used.

After going through the entire 72-bit ID, only one card—the card with the last "1" bit in its serial ID—is left. The configuration software then calculates an 8-bit checksum for the first 64 bits of the serial ID it read, and compares it to the received checksum. If the checksums don't match, the software restarts the isolation cycle.

With a single card with a valid serial ID now isolated, the configuration software issues the SET CSN command, giving the card its identity for future reference. Once given a non-zero Card Select Number, the card enters the Config state and will enter the Sleep state when the rest of the isolation protocol is performed on other PNP cards. The isolation protocol is repeated until there are no valid responses; at that point, all PNP devices are assumed to have been identified.

If there are no responses to the first iteration of the isolation protocol, the configuration software changes the READ_DATA port, assuming an I/O address conflict occurred. After attempting the first iteration of the protocol on all valid READ_DATA addresses with no response, it assumes no PNP devices are present.

Resource Allocation

Once every card has a unique Card Select Number set, card resource data can be read. A WAKE(CSN) command is issued to each card. Resource data is then read by polling the resource data status register and waiting for a resource data ready bit to be set. The resource data is read one byte at a time.

The resource data is stored in a series of "tagged" data structures. These structures are strings of bytes that contain the string length, a "tag" that identifies what type of data the string is, and the data itself. Resource data stored includes ANSI vendor/card ID strings, PNP version numbers, IRQ, DMA, I/O port, and memory requirements, and I/O dependencies as well as other significant configuration information.

The configuration software then uses all the collected configuration information to select the best configuration for the cards in the system. The software may interact with other parts of the operating system; for example, there may be resource configuration files that cause the software to force a card into a given address space. In any case, the configuration software then issues configuration commands to set the I/O ports, IRQs, DMA, and memory resources of each configurable PNP card.

There is an option within the PNP specification to have a non-configurable card. The benefit of this type of device is that the configuration software would at least have enough information from a non-configurable card's resource data to steer the configurable cards in a way to avoid conflicts.

Proliferating the PNP Specification

As can be seen from the description of PNP operation, software is critical to the success of PNP. While PNP can be supported entirely with I/O card drivers, this is not a desirable situation—for each card to have an independent PNP driver offers little in the way of resource management. A far better solution is to support PNP at the operating-system level. To this end, Microsoft intends to support PNP with the "Chicago" release of Windows that is planned for 1994. In the meantime, an incremental release of Windows may integrate PNP device drivers for network cards. At its PNP implementation seminar in June, Microsoft will also be offering some generic device driver code.

On the hardware side, PNP prototype cards exist

now, and commercially available cards are expected in just a few months. One of the design goals of PNP was to minimize incremental hardware costs to no more than five dollars per card.

Whether this goal is met depends on the sophistication of the card. PNP ISA cards require hardware that can disable the bus interface of the card. A PNP card must also recognize PNP interface addresses, so some additional address decoding is necessary. A configurable PNP card will need programmable base addresses for the ports and memory, and programmable IRQ line selection. Some added logic and memory is required to support the card ID process.

For manufacturers that currently have configurable ASIC-based cards, conversion to PNP support should involve little more than some extra gates—perhaps a small ROM—and another manufacturing turn to revise the ASIC. This entails additional design costs and, for most manufacturers, another ASIC NRE charge. Only those manufacturers with high volumes will be able to hold the incremental hardware costs to the target level.

For cards and manufacturers that aren't sophisticated, it may not be nearly as simple—a PNP interface controller will have to be added. While this could be implemented with ASICs or a handful of PLDs, at least two semiconductor manufacturers—Microsoft won't say which two—have expressed an intent to build PNP interface controllers. Such components would serve less sophisticated designers well, but it is unlikely that they would be offered at a price low enough to meet the five-dollar goal.

Conclusion

The benefits of PNP for new systems are obvious. It allows quick, efficient user upgrades of hardware with a minimum of hassles, making it an important differentiating point for system OEMs. PNP could also significantly reduce a company's technical support expenses, as there would be many fewer calls to the support line with configuration problems.

There are a few—albeit obscure—problems that need to be resolved, even for PNP-aware systems. There is nothing in the 0.9 specification, for example, to handle more than one hard disk controller or graphics adapter configured as a boot device.

PNP is also relying on the good will of hardware vendors, at least with the current specification. There is nothing in PNP's structure to prevent a manufacturer from taking an existing ISA board, adding a PNP resource definition that demands a given IRQ and memory space, and marketing it as PNP-ready. If there are enough boards available that demand specific settings, there will be a high probability of unresolvable resource conflicts between cards. The system won't crash, but the user probably will be prompted to pick which card gets to operate.

One unresolved issue is support for EISA, which already handles automatic configuration for EISA cards but not ISA cards. Given 12-bit I/O decoding and EISA configuration ports, a PNP card can be built to support configuration in both PNP-ISA systems and EISA systems. There is not widespread support for this proposal, so it's quite likely that the 1.0 PNP specification will make this an optional feature that board vendors can choose to support.

Compaq, the leading vendor of EISA systems, is clearly behind the push for EISA support. Compaq, joined by Microsoft, recently announced that it will lead the development of new PNP extensions to include EISA support. The ISA PNP specification, however, does not appear to be affected by the politics of EISA support. The EISA extensions may have been a move to enhance PNP and protect the current specification from a damaging controversy.

While older ISA systems without a PNP-aware BIOS can use PNP add-in cards, the benefits of PNP in this system are much lower than for new systems. Relatively simple PNP devices, such as internal modems, will probably configure automatically without problems. More complex, I/O-intensive hardware, such as high-end network adapters, will most likely require user interaction. In a typical scenario with a mixture of ISA and PNP cards, the installation software would identify locations of any obvious ISA cards (serial ports and graphics adapters, for example) and give the user an option to "lock out" certain addresses and interrupts. The installation software would then proceed with automatic configuration of the remaining PNP cards. Even this limited semi-automatic configurability is better than what currently exists in the ISA environment today.

On the surface, PNP may appear excessively complex when compared to other auto-configuring buses. This comparison is unfair. PNP successfully retrofits a feature-barren 12-year-old ISA bus with significant new features it was never intended to support. Hopefully, IC manufacturers will mask the complexity in standard cells for ASICs or standard interface chips.

For new ISA systems or old ISA systems that are completely retrofitted with PNP cards, the new protocol offers a significant improvement in ease of use, and could make user-installed hardware as hassle-free as it is on the Macintosh today. For ISA systems with old ISA cards, PNP offers some limited benefit in potentially reducing conflicts between newly added hardware and older non-PNP hardware. For many manufacturers, the addition of PNP will involve little increased cost. PNP looks like a promising method of upgrading an aging ISA standard to provide new benefits to PC users and manufacturers alike, but it is of little value without wide support from card vendors.♦

