COTS View

Case Study:
Space Station Robot Embeds Ada

Despite rumors of its demise, Ada is at the heart of the International Space Station's in-orbit Canadarm 2 where it assures software safety and reliability.

Robert Dewar, President,
Ada Core Technologies

While safety-critical characteristics have been introduced into the design of many programming languages, Ada is the language specifically targeted at "life-critical" systems. Developed between 1975 and 1984 by the US Department of Defense (DoD), Ada has been classically targeted for use in mission-critical embedded systems that emphasize safety, low cost, and a near-perfect degree of reliability. The most important safety features that make Ada ideal for development of fail-safe software include its information-hiding capability, its ability to provide re-useable code and its "strong typing", which helps detect and solve many types of coding errors at compile time, very early in the development cycle.

Despite the perception by some that Ada is a dying language, the fact is that Ada's use is on the rise and it's being adopted for some of the most rigorous and critical embedded applications under development today. Under contract to the Canadian Space Agency (CSA), MacDonald Dettwiler (MDA) chose open-source GNAT Ada 95 from Ada Core Technologies to develop control software for the Mobile Servicing System (MSS), an essential component of the International Space Station (ISS).

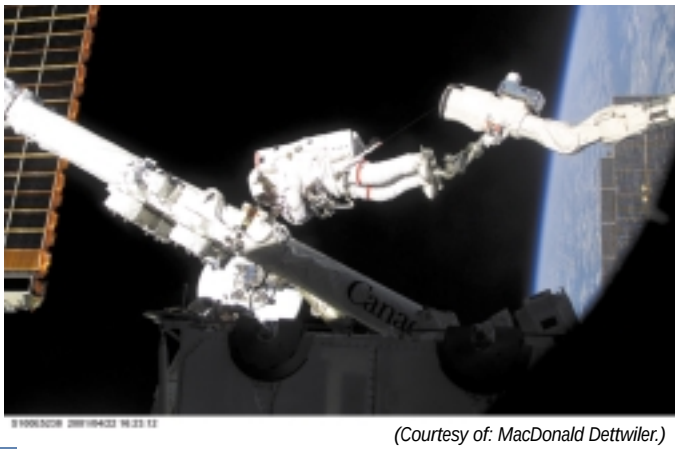
The MSS is a complex robotic manipulator system that plays a key role in space station assembly and maintenance. It helps move equipment and supplies around the station, supports astronauts working in space, and services instruments and other payloads attached to the space station (Figure 1).

Space-Based Robotic Arm

Ideal for a program like the MSS, Ada has clearly carved itself a comfortable and sustainable niche in large, complex high-reliability systems, including safety-critical systems where human life might be at stake. This language, which has little visibility compared with its cousins C and C++, continues be very effective in developing systems that absolutely must be reliable. So it is no surprise to find Ada in space—a harsh, unyielding environment where the slightest malfunction can lead to death.

The ISS-based, next-generation Canadarm 2, the key element of the MSS, is a bigger, better, smarter version of Canadarm, the robotic arm that operates from the cargo bay of the Space Shuttle. This arm is capable of handling large payloads and assisting with docking the space shuttle to the space station. The new arm, built specifically for the space station, is 17.6 meters (57.7 feet) long when fully extended and has seven motorized joints, each of which operates as a complex real-time embedded control system.

Canadarm 2 is "self-relocatable" and can move around the station's exterior like an inchworm. Each end of the arm is equipped with a specialized mechanism called a Latching End Effector that can lock its free end on one of many special fixtures, called Power Data Grapple Fixtures placed strategically around the station, and then detach its other end and pivot it forward. Unlike the original Canadarm, Canadarm 2 stays in space for its useful life, requiring a high-reliability design that allows astronauts to repair it in-orbit. It nearly goes without saying that the reliability of Canadarm 2 must be unimpeachable, based on high-integrity software and



(Courtesy of: MacDonald Dettwiler.)

Figure 1

Canadian Astronaut Chris A. Hadfield stands on the end of the Space Shuttle's Canadarm while bolting Canadarm 2 together.

(Courtesy of: MD Robotics.)

Figure 2

The Mobile Servicing System's Robotic Work Station (RWS) features a display and control panel, hand controllers, video monitors and computers to provide a highly reliable, seamless interface between man and machine.

safety-critical design constructs that need to be formally and rigorously specified, designed and implemented.

Safety-critical software systems like Canadarm 2 often employ multiple levels of safety criticality. Historically, these have been deployed on systems where each function executes on a dedicated processor. The need to lower maintenance costs and reduce the size, weight, and power consumption of older embedded computer systems, combined with the availability of modern processor technology, has created the demand for commercial run-time systems.

Ada Core's GNAT High-Integrity Edition no-run-time instantiation of Ada 95 was used to develop Canadarm 2, allowing processing tasks with multiple threads of activity, as well as coordination and conformance of multiple processors communicating over well-defined interfaces. GNAT also facilitates a software architecture and implementation that is seamlessly extensible, allowing the integration of future phases of the robotic system.

The main Ada target processor on the Canadarm 2 is an embedded Intel386 device running on a proprietary board that sits at the heart of the arm's Robotic Work Station (RWS). The RWS controls the seven-joint servo mechanisms on the arm, sending commands to motor-control microprocessors, and closing the control loop by reading feedback from position sensors (Figure 2).

The RWS computer also handles the task of scanning user-interface inputs that allow astronauts to control the arm. Software for each joint-control processor is also written in Ada. I/O communications between the RWS and the joint processors occurs over MIL-STD-1553, known as the Aircraft Internal Time-Division Command/Response Multiplex Data Bus (handled by one processor).

Performance Legacy

While the concept of mission-critical systems has been widely adopted throughout the computing industry, it was the DoD that created the original concept. Ada, named after Ada Byron, founder of scientific computing, was born out of the DoD's mid-1970s concern over the proliferation of programming languages. The DoD

advocated a single language, based on solid software engineering principles. Following its 1983 unveiling, the DoD mandated Ada as the official language in 1986. Ada's demise was widely predicted after the DoD removed the mandate in 1997. However, Ada's use has actually risen since then. Ada increasingly has its place as a model of software reliability, reusability, readability, and portability among embedded systems developers.

Ada was the first mainstream programming language to incorporate constructs for multi-tasking and real-time programming, with well-defined interactions between complex features such as tasking and exceptions. It also includes asynchronous transfer of control, protected records, better user access to scheduling primitives, additional forms of delay statements, and parameterized tasks.

In the 1980s, Ada consistently outperformed established programming languages like Pascal, Fortran, and C. In the 1990s, Ada continued to surpass C++ in performance evaluations measuring capability, efficiency, maintenance, risk, and lifecycle cost. In a 1993-94 study of the 10-year history of Verdix Corporation's use of Ada and C development, preliminary findings indicated that Ada was twice as cost effective as C, said Stephen F Zeigler, Ph.D. in his document "Comparing Development Costs of C and Ada".

In-depth analysis of Verdix's general development methodology ultimately showed that when the effects of makefiles and of external costs were factored in, Ada costs were on the order of half the cost of "carefully crafted" C code. The simplest reason for this is that Ada inherently encourages developers to spend more time notating their code in writing, communicating information to future readers of that code. The benefits derived from this include:

Improved Error Locality. Ada bugs are often indicated directly by the compiler, giving developers fewer places to look for bugs. Bugs are often discovered very soon after they are created, and are considered "local in time." If a bug is easily identified as being close to the place in the code where it actually occurs, it is considered "local in space." Ada compilers are good at both time and space locality for bugs, greatly reducing development time.

Better Tool Support. Ada development intrinsically causes a great deal of information to be sent to the development tools. This characteristic is most pronounced for tasking, where the Ada tools can create parallel, distributed multiprocessing versions of ordinary-looking programs without developers having to do much more than they did for a single processor. Another big win is in machine code, where users can get free access to the underlying hardware without having to give up the semantic checks and supports of the high level language.

Improved program design. One of Ada's more subtle effects is to encourage better program design, avoiding the "quick-fix" habits unfortunately common to C programming. Examples include: C allows users to create a global variable without registering it in a ".h" file; C allows users to avoid strong typing easily. The effects of such details are subtle, but can account for some of the "progressive design deterioration" effects that can lead to many extra hours of debugging and interpreting old code.

Metric	Reasons
Availability/Reliability	Ada is highly reliable because its compilers rigorously check the code at compile-time, enabling the programmer to locate and remove defects early in the programming process.
Code Size	First generation Ada compilers produced large executables, and the stigma of large executables stuck with the language. Today's optimizing Ada compilers produce the tightest embedded system code available.
Isolation	Ada effectively compartmentalizes code at run-time, eliminating unpredictable interactions between code modules.
Portability	Ada code is easily portable across microprocessor architectures, making hardware migration and upgrades cheaper and faster.
Readability	Ada code is inherently very readable; errors are easy to locate and correct before compile-time.
Reusability	Ada uses a modularized structure with generic procedures and data abstractions. The result is exceedingly reusable software components, reducing costs for each new project. NASA did a systematic assessment that measured between 60 percent and 80 percent Ada code reuse from one satellite system to the next. This was comparing reuse against past use of Fortran, which was NASA's primary language for satellite systems. With Fortran, code reuse was about 25 percent to 30 percent.
Verifiability	All Ada compilers verify code against the Ada standard using the Ada Conformance Assessment Test Suite (ACATS formerly known as the ACVC tests).

Table 1

A list of factors making Ada increasingly popular in military and high-availability commercial systems.

Ada Embedded

With embedded systems residing in a variety of applications ranging from satellite and telecommunication systems to networking routers and passenger aircraft, an increasing number of lives and dollars now depend on the reliability of embedded systems software. Joint Strike Fighter, various armored vehicle programs, the Apache helicopter—dozens of programs, maybe more—are still actively developing in Ada and spending millions of dollars. Boeing's choice of Ada for high-profile commercial projects like the Boeing 777 is a good example. The 777 is an all-Ada plane except for the entertainment system which is coded in C++.

Penetrating markets long dominated by C++, Ada is surfacing outside the traditional realm of aviation and aerospace, in applications as diverse as meteorological imaging and yachting security. Compaq, for example, used Ada to implement its IN7 project, first on SCO Unix, then on Tru64 Unix v5.1, replacing DECada. IN7 is a relatively complex code mixing several languages and compilation tools, and one of the major difficulties was to get the generated code to run correctly.

IN7, one of the major implementations of the telephone central office switch Signaling System Number 7 (SS7) standard, needed to be fully distributed across multi-computer systems for high availability and high performance to telecom equipment manufacturers and software developers. Compaq turned to Ada to meet these demands, and GNAT specifically because it is targeted at many platforms supported by IN7. While Ada's primary advantage is its support for open systems and interoperability, Table 1 shows several other factors also contribute to higher productivity and lower cost.

Ada's Lifecycle

Research has shown that between 60 and 80 percent of software costs occur in code maintenance—after a system is actually developed and implemented. MacDonald Dettwiler knew these post-project costs could be even higher for the space-based Canadarm 2. Thus, the reliability and long-term robustness of the Ada code took on a very real economic incentive as the MDA engineers considered project lifecycle costs that went far beyond issues pertaining to the development phase of the project.

But will Ada survive? Despite the steady increase in Ada use for safety-critical embedded systems, inaccurate perceptions still linger about this stalwart language. Some programmers remain convinced that Ada will eventually fade away without its DoD mandate.

But, the truth is that Ada is not going to go away. Instead, it will thrive as a language-of-choice for the most demanding life-critical applications. Ada's founding principle was based on rigorous software engineering practices, and Ada is still the only internationally standardized programming language specifically designed for large, complex applications. For real-time, safety-critical applications Ada is still the undisputed leader. ▮▮

Ada Core Technologies
New York, NY.
(212) 620 7300.
[www.gnat.com].

COTS View

Case Study: Designing a Pentium-Based Product for Extreme Environments

Running PC applications in harsh environments requires Pentium single-board computers designed for that purpose—even though PC chipsets weren't.

George Schreck, V.P. Engineering,
Thales Computers

To take advantage of available software that runs on the Pentium processor, severe environment system designers must incorporate a number of features that match the PC's architecture. The operation of the system software relies on long-established peripherals that fit into the memory map in the same locations that they have been in since the PC was established. Additionally, for cost and real estate savings reasons, it seems to make sense to take advantage of the large-scale integration components that have been designed for PCs and are available in abundant supply at low cost.

The trouble is, these components weren't designed for harsh environments, extended temperatures or ultra long-life reliability in defense applications. Instead, harsh environment system designers have to make modifications to their designs—including testing of the components themselves—before Pentium single-board computers will work reliably. Furthermore, after taking these steps, quantitative testing must prove a successful result.

Matching Components to the Environment

Intel, as well as other vendors, produces commercial temperature range chip sets that contain much of the processor interface, as well as the actual standard PC peripheral interfaces. One such Intel chip set is the 440BX that consists of the 82443BX and the 82371AB chips. The 82443BX chip contains a processor interface, AGP controller, PCI controller, DRAM controller and power management support. One of these chips, the 82371AB PIIX4, contains a PCI to ISA bridge, a DMA controller, an IDE controller, an interrupt controller, a USB interface and a number of other functions.

Figure 1

This example of a conduction-cooled chassis is a Thales Computers ATR-style enclosure that features a totally sealed design and highly efficient conduction heat transfer path.

These two components make up the majority of the core PC bus interfaces and peripherals. To complete the common function set of a PC, an I/O controller such as the Standard Microsystems Corporation FDC37B787 is available. That particular chip contains functions such as: two serial ports, a parallel port, a keyboard interface, a floppy disk controller and a real-time clock.

But here's where the first harsh environment design obstacle appears. Many of the chip sets that are available with the higher clock speeds are not available in extended temperature.

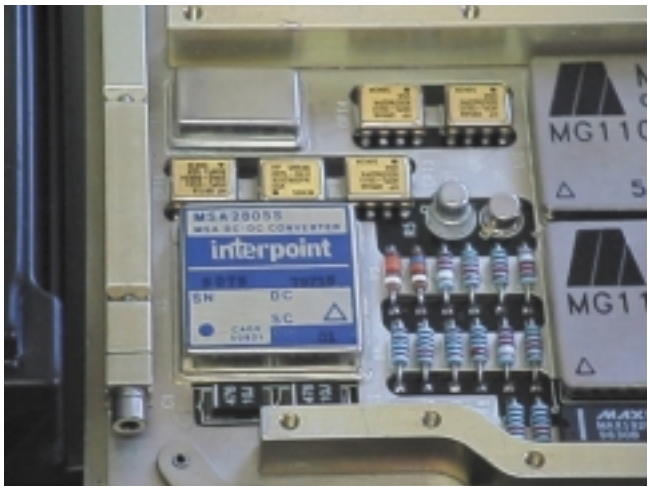


Figure 2

A metal cooling layer beneath the components works well for components that have more surface contact area. The cooling metal layer is easily visible under the resistors.

Because of the issues of the complexity of PC BIOS code and legacy hardware, departing from standard components can cause software incompatibilities. For designs that require lower performance, Intel has provided a small selection of Pentium processor chips and peripheral components that are rated for a wide temperature range and are known to work well together.

The Intel "Extended Temperature Pentium Processor with MMX Technology" is rated for -40°C to 115°C case temperature and is available up to 166 MHz. A complementary chip set is the Intel "Extended Temperature 430TX PCIset (430TX)" consisting of the Extended Temperature 82439TX System Controller and the Extended Temperature 82371EB PCI ISA IDE Xcelerator. The chip set has also been specified for an extended temperature range of -40°C to 115°C.

But if a design requires more performance, it may be necessary to use parts that are not specified for wide temperature operation by the vendor (for example, Intel or Standard Microsystems). When using these non-traditional parts, it is necessary to understand their technology well enough to have confidence that they can be made to run over extended temperature. Extended operation of those parts can then be guaranteed by testing them to the new design requirements.

To maintain the health of a component, its junction temperature must stay within its design range. To control junction temperature, it is necessary to devise methods to remove heat from the components with adequate efficiency to keep them cool enough to make them reliable. Running a component at the top of the junction temperature limit will cause its failure rate to be increased and the component to be seriously degraded. The junction temperature of a component is a function of its power dissipation and the thermal resistance of the

heat removal path. For an air-cooled component the general equation is:

$$T_j = T_a + P\ (R_{\theta jc} + R_{\theta ca})$$

T_j junction temperature
T_a ambient temperature
P power consumption
$R_{\theta jc}$ thermal resistance of the junction to component case
$R_{\theta ja}$ thermal resistance of the case to the air

For certain designs, it will be necessary to use components that may not operate reliably when exposed to very cold temperatures. In those cases, it may be necessary to add some heat to the component. This is a practice that has been used on space vehicles for some time. For some systems, the application of heat is done at a whole board level using fairly large radiant or conduction heaters. However, for individual components, there are commercially available resistive heaters and controllers from companies like Minco Products. Alternately, it is not too difficult to design a heater specific to the application. To complete the package, there are digital thermostat chips available to control the heaters that also could be used to hold a circuit in reset until prescribed temperature requirements are met.

Thermal Management Details

For the most severe environments, conduction cooling is typically employed instead of air-cooling via convection (Figure 1). This cooling method is preferred because the electronics can be totally sealed from the environment and are therefore not exposed to contaminants. Additionally, the temperature of the components can be better controlled because of a better heat transfer path.

Historically, a high percentage of the conduction-cooled products being designed had a metal cooling layer added to the top of the printed circuit board before the components were added. The metal was generally aluminum or copper and had holes machined in it to clear the leads of the components (Figure 2). This method worked well for DIP parts that had a lot of surface contact area. However, it does not offer a good solution for surface mount components.

Early printed circuit boards, designed to accommodate both DIPs and surface mount components, had a large heat sink, or core, in the center of the printed circuit board. The heat sink was a piece of copper that was 0.020" thick or more. The copper was designed as part of the circuit board design. The thickness of the copper was established by the heat transfer characteristics that were required. To get the heat from the components to the core, small pads of copper were etched into the design beneath the components. Thermal vias were then placed between the core and the conduction pad.

To facilitate heat removal, the parts then had thermal transfer material, like a thermal transfer epoxy, put between them and the thermal pad. The copper core was exposed near

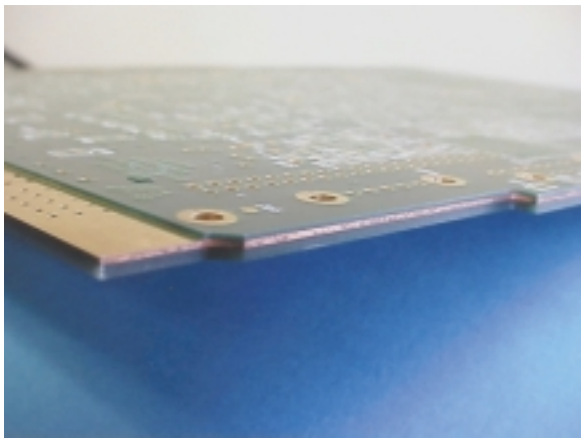


Figure 3

Exposed copper near the edge of a solid-copper core (0.020" thick) board allows for cooling of the board's core. Heat from the components transfers conductively to the core and is then carried to the core's edge.

the edge of the board to allow a place to cool the core (Figure 3). Although this design worked adequately for cooling many surface mount components, it is very difficult and expensive to fabricate.

One of the most cost-effective designs for removing heat through the board involves using multiple thin core layers. The primary advantage is reduced cost. Since the layers are thin, the PCB manufacturers treat them like power planes, and nothing special needs to be done to handle them (Figure 4). As with the previous method, thermal pads and vias are employed to remove the heat from the parts and move it into the core layers. The primary disadvantage to this system is the additional cost that is incurred from adding more layers to the boards.

The most common method of heat removal is through an "over the top" heat sink (Figure 5). This method involves manufacturing a heat sink that complies with the surface of the components. A thermal transfer compound is applied between the heat sink and the components to guarantee contact and to make up for differences in coefficients in thermal expansion. The advantage to this is that the board can be assembled without any attention to thermal considerations until the product is through initial test stage. The major problems with this approach are the additional costs associated with machining a separate heat sink and the assembly cost of adding it to the board.

Real World Example

The Thales Computers PENTX board was designed using many of the tenets previously described. Because of the high power consumption of the Pentium processor relative to the board size, the board uses a combination of multi-layer conduction-cooled construction with an external heat sink used as a supplementary heat transfer path (like that in Figure 5). Much of the power dissipation of the Pentium chip, the clock

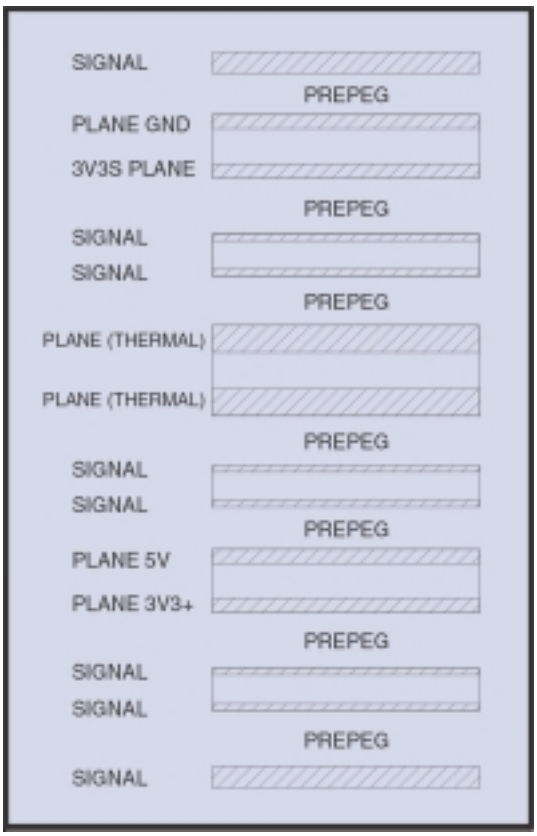


Figure 4

Using multiple thin core layers is one of the most cost-effective methods for cooling components. This PCB layer stack up uses two dedicated copper cooling layers.

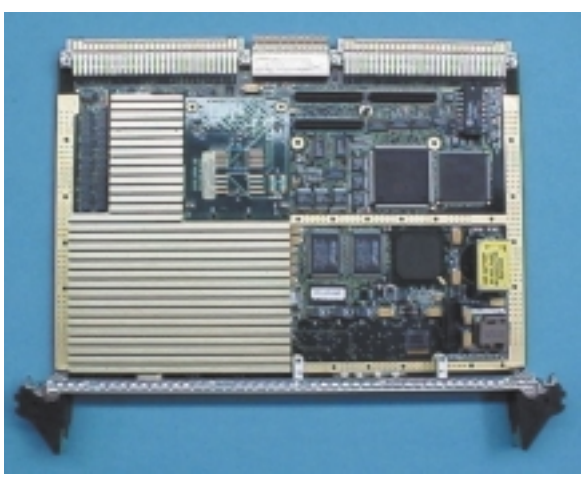


Figure 5

The most common method of heat removal, as seen on the convection cooled version of the Thales Computers PENTX3 board, is the "over the top" heat sink allowing the board to be assembled and tested before the thermal transfer compound and heat sink are added.

Figure 6

Conduction cooled PENTX board with thermocouples attached. Box covers are removed for clarity. The colored dots represent the approximate location of the thermocouples under the heat sink.

Phased Locked Loop (PLL) chip and the VMEbus interface chip is handled by the external heat sink. The printed circuit board is utilized to cool many of the other components.

As mentioned previously, several components that support the Pentium are not available in extended temperature. To meet the design requirements, a combination of pre-screening and screening on-board is used to validate parts. Even with careful attention to the details within the design, it was still desirable to add a silicon supplemental heater under the PIIX4 chip on the board. It was discovered that the part would sometimes fail to generate refresh cycles to the memory at temperatures approaching -47°C. The heater is designed to come on below -35°C. The thermostat part doubles as an on-board thermometer for the user's application code.

For testing, a number of thermocouples were attached to the PENTX board at strategic locations and the board was loaded into a conduction-cooled test chassis. The thermocouples were located in the following positions: on the surface of the Pentium chip, on the 82443BX chip, on the Universe VMEbus interface chip, on the PLL clock generator, on the outside wall (cold wall) of the test enclosure and above the board for ambient air temperature (Figure 6). Those par-

Cold Temperature Test Chart

Reading #	TIME (Minutes)	TEMPERATURE (°C) Chamber	Processor	Northbridge	Universe	Clock Gen	ColdWall
1	28	1.0	11.6	11.5	11.5	11.3	8.2
2	44	-14.0	3.8	-4.0	-4.0	-4.2	-7.0
3	62	-27.6	-19.1	-19.1	-19.2	-19.3	-22.2
4	79	-37.6	-31.7	-31.8	-32.0	-32.1	-34.0
5	97	-43.9	-40.9	-41.0	-41.0	-41.1	-42.2
6	112	-44.4	-43.9	-44.0	-44.1	-44.1	-44.3
7	117	-44.5	-44.2	-44.2	-44.2	-44.2	-44.4
8	119	-44.6	-37.8	-41.1	-40.5	-41.0	-44.3
9	121	-44.5	-39.1	-39.3	-40.2	-40.2	-44.2
10	124	-35.0	-33.1	-36.1	-36.0	-36.4	-40.8
11	127	-25.0	-29.2	-32.2	-32.2	-32.4	-36.0
12	130	-15.0	-21.6	-24.5	-24.7	-24.9	-27.6
13	135	-5.0	-12.9	-15.7	-15.8	-16.1	-18.3
14	140	5.0	-2.8	-5.5	-5.6	-5.9	-8.1
15	145	15.0	7.4	4.8	4.7	4.4	2.0
16	150	25.0	17.3	14.7	14.7	14.5	11.9

Figure 7

The Cold Temperature Test Chart represents the data collected for this portion of the test.

ticular components were chosen because of their rated high-power consumption and the resulting need to keep them cool.

The entire assembly was located in the environmental chamber and the assembly was cooled with the power off to approximately –45°C. A temperature chart for each of the locations verses time was generated (Figure 7). When the chamber was down to the target temperature and all the components were stabilized at that temperature, the board was cold started and allowed to run in a continuous diagnostic loop. The chamber was then set to start a temperature increase to a target temperature of +25°C.

The board temperature was close enough to stabilized for the purpose of this experiment at Reading Number 6. Notice that all the readings for all the thermocouples were within 0.5 degrees. The power was then switched on at Reading Number 7. That can be seen by the increase in temperature of the processor and was also noted by the test technician monitoring the video display. Since the board successfully cold started, it was deemed successful for that portion of the test.

The chamber target temperature was subsequently changed at Reading Number 7 to +25°C. Readings were then taken periodically to track the effectiveness of the heat transfer system on the board. Notice that the board was never allowed to stabilize at +25°C. It was not necessary because the difference in temperature between the components and the cold wall were the interesting data points and the chamber was only being used to facilitate those measurements. If one graphs the data, some of the thermal characteristics of the board become more obvious (Figure 8).

During the cool-down part of the cycle, the processor chip, having the longest heat transfer path and one of the most thermal masses of all the components, is the last thing to temperature stabilize (Figure 9). It always lags the other components during the cool-down. Also, notice that since the processor generates much of the heat on the board, it takes a rather large leap as soon as power is applied. During the warm-up part of the cycle, the largest difference between the

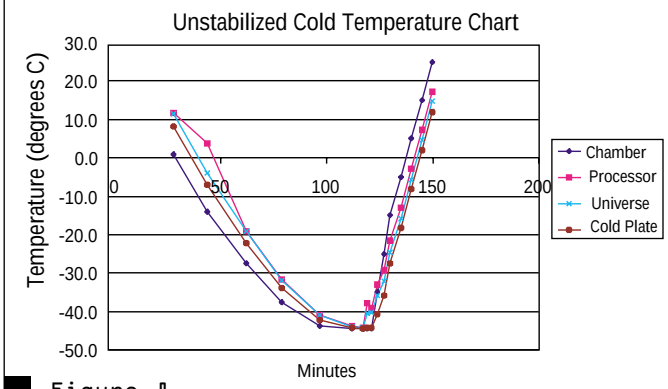


Figure 8

The temperature plot of various components on the PENTX board show the effectiveness of the cooling methods employed at cold temperatures.

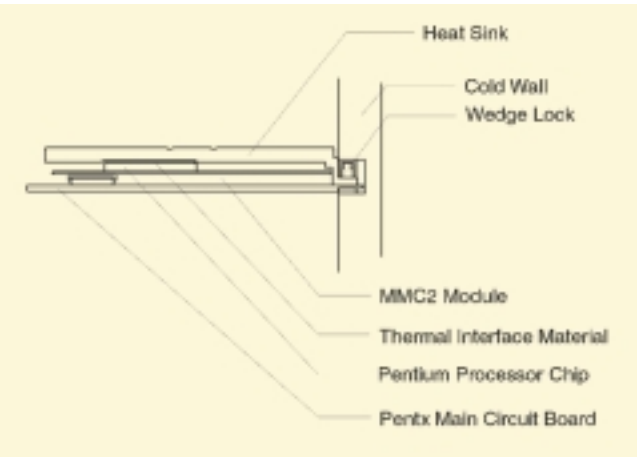


Figure 9

Thermal transfer path of the Pentium processor chip to the cold wall of the box. Heat is conducted from the processor to thermal interface compound into the heatsink, through the wedge lock or the circuit board and out to the cold wall.

Hot Temperature Test Chart

TIME @ temp Reading #	(Minutes)	TEMPERATURE (°C) Chamber	Processor	Northbridge	Universe	Clock Gen	ColdWall
1	40	25.0	38.1	35.7	35.6	35.2	26.0
2	27	35.0	49.2	46.9	46.8	46.3	36.2
3	30	45.0	59.2	56.8	56.7	56.2	46.0
4	33	55.0	69.1	66.7	66.7	66.1	55.9
5	62	70.0	84.2	82.0	81.9	81.3	71.6

Figure 10

The High Temperature Chart shows a +12 to +13°C difference between the hottest component and the cold wall.

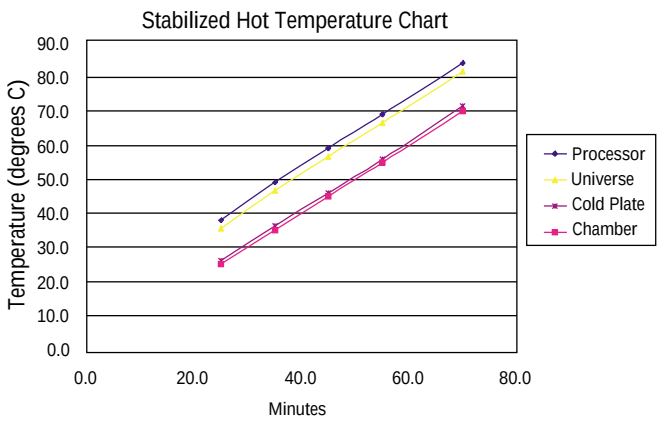


Figure 11

Temperature plot of various components on the PENTX board showing the effectiveness of the cooling methods employed at hot temperatures.

cold wall temperature and any component during the warm-up is +7.7°C, which was well within the design requirements for the board.

The hot temperature testing was done a little differently (Figure 10). Since the objective on the hot side is to make sure each of the components is being adequately cooled, the same setup was used, but the temperature of the system was allowed to stabilize at a number of predefined chamber temperatures. The maximum chamber temperature was chosen to yield a target cold wall temperature of +71°C to match with a military specification.

It is important to notice that, for a stabilized system, the difference in temperature between the hottest component and the cold wall remains consistently around +13°C. Although the extrapolation is not completely linear, the implication is that if the cold wall temperature were +78°C, the case temperature of the Pentium chip would reach the maximum desired +90°C (Figure 11). The other implication is that, if the system were cooled to a cold wall temperature of –45°C while running, the Pentium processor chip would remain above –35°C. ■■

Thales Computers
Raleigh, NC.
(919) 231 8000.
[www.thalescomputers.com].