Published in the ITE Journal, May 2007

The ATC Application Programming Interface – Closing the Technology Gap

Ralph W. Boaz and Douglas Tarico

Abstract. There has never been more demand for technological power for on-street transportation equipment. Unfortunately, the majority of the fundamental equipment in use today is grossly underpowered for many Intelligent Transportation Systems (ITS) applications creating a technology gap that must be remedied by adding additional devices to already crowded transportation cabinets. Alternatively, the Advanced Transportation Controller (ATC) Controller and Application Programming Interface (API) Standards provide a powerful and flexible platform through which complex solutions can be developed and deployed. This article provides an overview of these standards with special emphasis on the features of the API Standard and the opportunity it provides.

Keywords. API, ATC, interoperability, 2070, NEMA

THE NEW DILEMMA ZONE

The world has seen dramatic advances in technology over the last 25 years. Unfortunately, the transportation community has not had the same advancements in its fundamental field equipment. Consider the comparison in Table 1. While PCs today are 7,000 times faster than they were in 1982, traffic controllers today are only 22 times faster then they were in 1982. This represents a relatively modest improvement in capability at best. Conversely, the demand for capability has never been greater. Common traffic applications such as automated toll collection, video surveillance, video detection, red light enforcement, and dynamic message signs have created a technology gap that has been crudely bridged by adding more "boxes" (separate devices dedicated to a specific function) to the transportation cabinet. This box-level expansion approach leads to the complaint that can be heard from almost every traffic engineer in the country, "There isn't any more room in the cabinet." The technology gap is even more exaggerated when one considers applications for advanced corridor management in catastrophic environmental conditions or homeland security applications to help combat terrorism. Figure 1 illustrates this problem.

Table 1. Comparative change in performance between PCs and traffic controllers.

Year	PC	MIPS*	Traffic Controller	MIPS*
1982	80286 (6 MHz)	1	170 (1.0 MHz)	0.2
2006	Pentium 4 (3.6 GHz)	7,000	2070 (25 MHz)	4.5

* – Millions of Instructions Per Second (MIPS) estimated from published data.

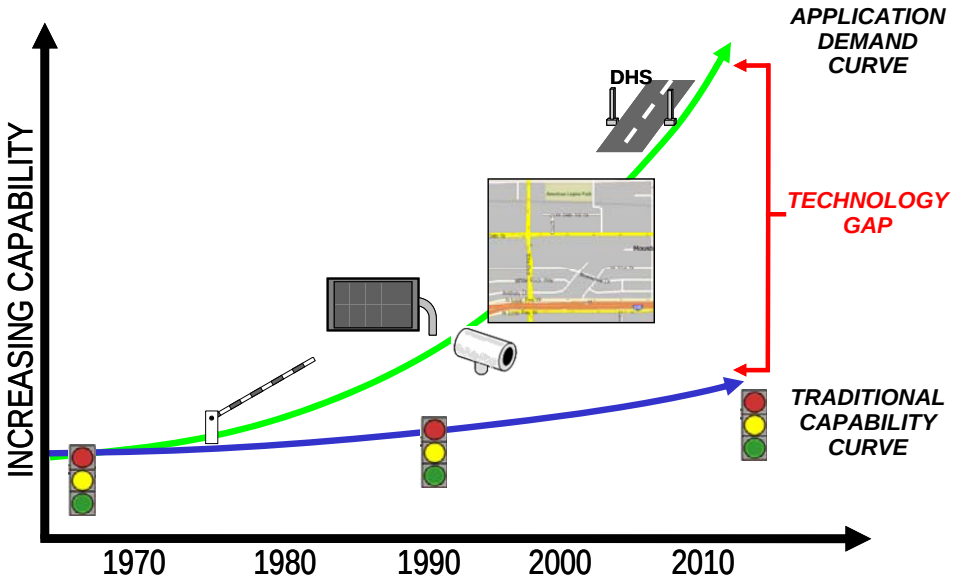

Figure 1. The technology gap in transportation equipment.

OVERVIEW OF THE ATC CONTROLLER AND API STANDARDS

In the mid 1990s, the ATC Joint Committee (JC) was formed in an effort to close the technology gap by creating standards that are "open" (the specification is publicly available), use more contemporary designs, meet the needs of current transportation applications, and can grow with technology. The ATC JC is made up of representatives from the American Association of State Highway and Transportation Officials (AASHTO), the Institute of Transportation Engineers (ITE), and the National Electrical Manufacturers Association (NEMA). There are four ATC Standards: the ATC/2070 Standard,[1] the Intelligent Transportation Systems (ITS) Cabinet Standard,[2] the ATC Controller Standard,[3] and the ATC Application Programming Interface (API) Standard.[4] The technical work on the standards is carried out by three Working Groups (WGs) under the oversight of the ATC JC: the ITS Cabinet WG, the ATC Controller WG, and the ATC API WG.

ATC Controller Standard

Central to the idea of closing the technology gap, are the ATC Controller Standard and the ATC API Standard. They work together to provide a flexible technology platform that can be incorporated in innumerable controller designs and cabinet architectures. The ATC Controller Standard specifies a controller architecture where the computational components reside on a single small (5" x 4") printed circuit board (PCB), called the "Engine Board," with standardized connectors and pinout. It is made up of a central processing unit (CPU), a Linux operating system (OS), memory, external and internal interfaces, and other associated hardware necessary to create an embedded transportation computing platform. The Engine Board plugs into a "Host Module" which supplies power and physical connection to the input/output (I/O) facilities of the controller. While the interface to the Engine Board is completely specified, the Host Module

may be of various shapes and sizes to accommodate controllers of various designs including both 2070 and NEMA controllers. Both the ATC/2070 Standard and the Caltrans Transportation Electrical Equipment Specifications (TEES) are being updated to include options for the Engine Board. NEMA-compliant ATC controllers are already being deployed using the Engine Board. This concept also allows more powerful Engine Boards to be deployed in the future without changing the overall controller and cabinet architecture.

The Engine Board runs a powerful operating system called Linux that is rapidly gaining popularity within the embedded computing industry. Linux offers many benefits: it is available from numerous OS vendors, the source code is in the public domain, it is supported by every major computer processor manufacturer and, generally, there are no runtime license fees for embedded systems. Because the Linux OS configuration is specified in the ATC Controller Standard, software developers can write application programs that will operate on any ATC controller unit no matter who manufactures it. It can be said that the ATC Controller Standard facilitates *interchangeability* of software on different ATC platforms. The Linux OS also provides the basic capability for running multiple programs concurrently on the same controller unit. However, this capability is not complete without the ATC API software as described in the next subsection.

ATC API Standard

ATC API Standard defines a software interface that, when combined with the Linux OS specified in the ATC Controller Standard, forms a universal interface for application programs to operate on all ATC controller units. This allows application writers to develop innovative software solutions for ATC controller units regardless of the controller manufacturer. While the ATC Controller Standard facilitates *interchangeability* of application programs between controllers from different manufacturers, the ATC API Standard supports *interoperability* of multiple application programs from different software vendors on a single controller unit. The API Standard enables interoperability by defining software interfaces which allow application programs to share the ATC controller unit's front panel and field I/O systems. Using the ATC Controller and API Standards together enables future advances in processing power to be applied to deployed ATC controllers while retaining the ability to operate the software applications of the existing transportation system. The API Standard provides for application software portability at the source code level. The application software source code may need to be recompiled to operate on different Engine Boards. While this is not as direct as the binary software portability provided by the Model 170 or ATC/2070 platforms, source code portability provides design freedom for the Engine Board manufacturers and allows Engine Boards to evolve and incorporate new technologies over time.

Figure 2 shows the API Front Panel Manager Window which is used to select between the user interfaces of multiple application programs. In this example, there are three programs running on the controller: a camera controller program, a traffic signal program, and a radiation detection program. The user interface to the different programs may be selected by simply pressing the associated key from the front panel keypad. The asterisk (*) next to the program name indicates that the user interface for that program will be displayed when the controller powers up. This is assignable by the person operating the controller. The Front Panel Manager Window can be

displayed at any time by pressing a double asterisk (* twice within a 1.0 second time period) followed by an escape key (ESC). Pressing the NEXT key in the Front Panel Manager Window causes a Configuration Window to be displayed which provides system information about the controller.

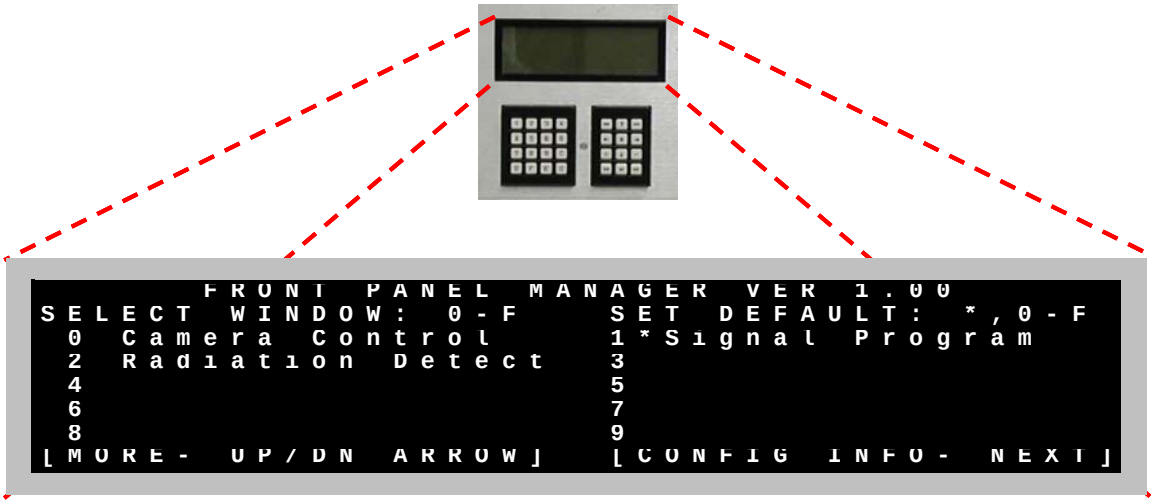

Figure 2. The API Front Panel Manager Window showing three application programs.

The API Front Panel Manager is designed to work with various front panels possible for an ATC such as the controller's fixed front panel, a laptop computer or a handheld device. If the size of the controller's physical display changes, the Front Panel Manger Window will automatically change size. Users may use arrow keys to scroll the screen up and down if there are too many programs to be displayed on one screen. Figure 3 shows the user interfaces of the three programs running in this example. It should be emphasized that the application programs provide their own specific user interfaces. The ATC API software provides the capability to operate the programs concurrently and switch between their user interfaces.

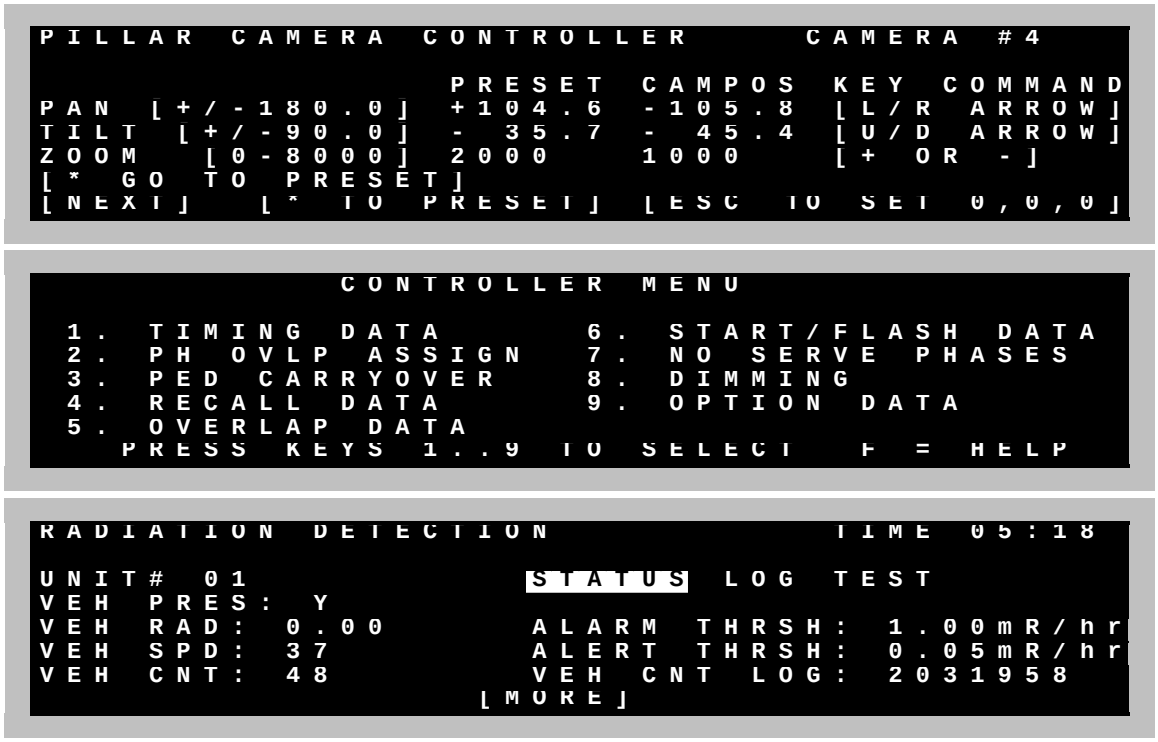

Figure 3. Example user interfaces for three programs on a single controller unit using the ATC API Front Panel Manager (only one screen displays at a time).

For the programs in a traffic controller to be able to interact with a device outside of the cabinet, there must be a connection between the controller and that device. Typically, this is done through a field I/O interface between the controller and cabinet. All of the major transportation cabinet architectures in use today have a similar concept of inputs (usually detectors) and outputs (usually load switches) addressable as “points” or “pins” to a single program in the controller. The API Standard expands this concept via the API Field I/O Manager by allowing all currently running programs to read all I/O points available to the controller but require an exclusive assignment of an output point to a single program. Figure 4 illustrates an ITS Cabinet with three programs with each program assigned to different load switches in the cabinet. The API simplifies field I/O functions by providing a single common internal software interface for all of the current cabinet architectures including the Caltrans Model 332 Cabinets, NEMA TS 1 and TS 2 Cabinets, and ITS Cabinets. This allows software developers to write their application programs in a consistent manner regardless of the target cabinet architecture.

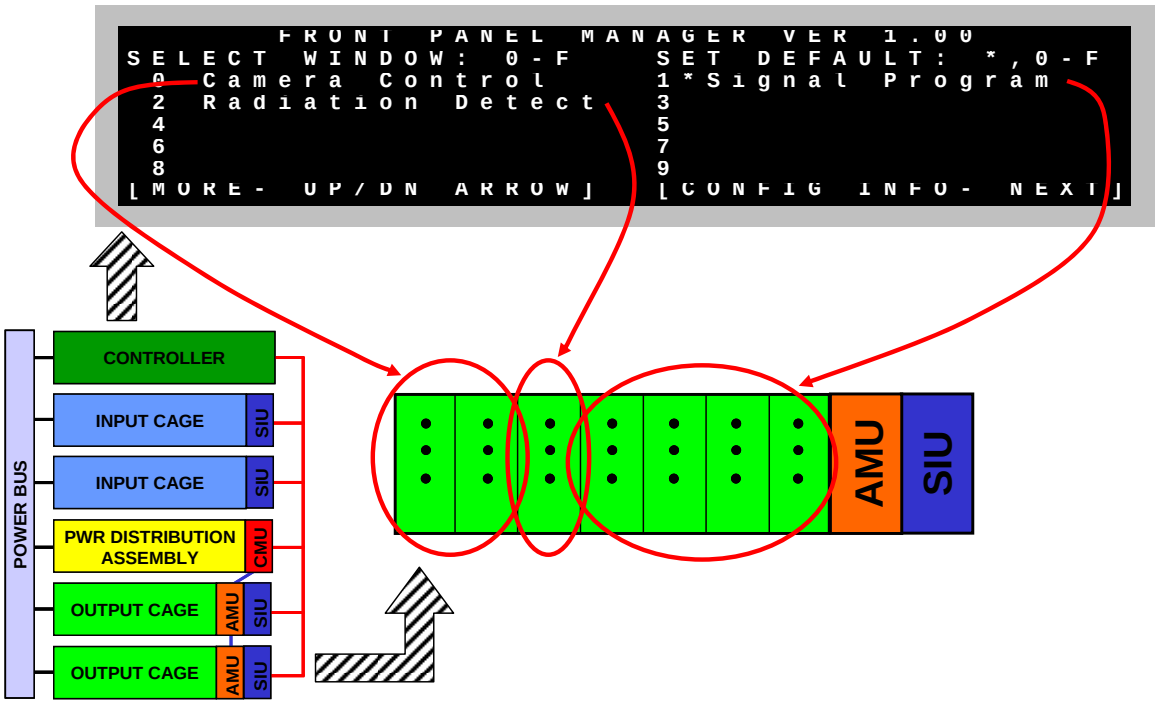

Figure 4. The API Field I/O Manager allows concurrently running programs to be assigned to separate output points in the cabinet.

HOW THE ATC STANDARDS WORK TOGETHER

Figure 5 illustrates the organization and layered architecture of the ATC software. The "Linux OS and Device Drivers" reflects the specification of the Linux operating system defined in the ATC Controller Standard. This includes functions for things typical in any computer system such as file I/O, serial I/O, interprocess communication, and process scheduling. It also includes the specification of the device drivers necessary for the Linux OS to operate on the ATC hardware. "API" refers to the Front Panel Manager and Field I/O Manager systems discussed in the previous section. As shown in Figure 5, both users and application programs use the API to interface to ATC controller units.

The division of the ATC software into layers helps to insure consistent behavior of the software environment between ATC implementations and also provides a migration path to new ATCs in the future. The relationship between the Hardware Layer and ATC Board Support Package (BSP) Layer is maintained, for the most part, by the worldwide Linux user community. There are strong market incentives for Linux developers to maintain the Linux standard and insure consistent functionality of Linux across multiple hardware platforms. The relationship between the ATC BSP Layer and the API Software Layer is maintained by the transportation community. Functions in the API Software Layer access the controller unit through the functions in the ATC BSP Layer. If the programs in the Application Layer only reference the controller unit through the API Software and ATC BSP Layers, they will be interchangeable between different ATC controllers units. The source code may need to be recompiled to accommodate differences in the

underlying BSP and Hardware Layers but this is much simpler than porting software which interfaces directly to the Hardware Layer.

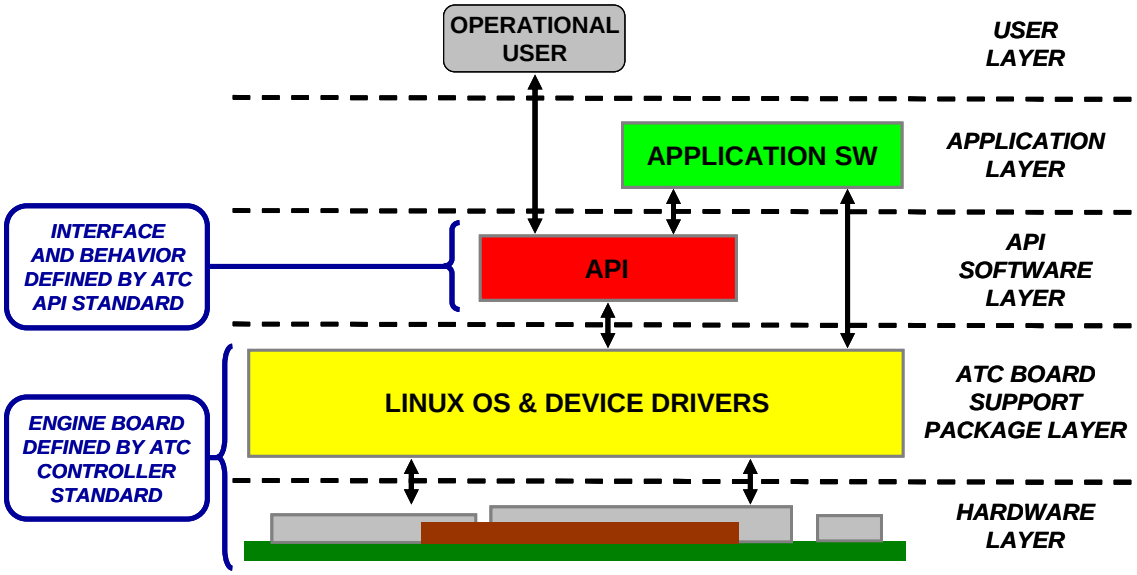

Figure 5. ATC layered software architecture.

AN OPPORTUNITY

The ATC Controller Standard was approved in 2006 and controllers are already being deployed which meet the standard. The API Standard is at a user comment stage and is scheduled to be published in the summer of 2007. After publication, the API will be deployed as agencies require adherence to the standard in their specifications. Manufacturers will implement the API Standard to meet the demand of the agencies. This deployment approach tends to be slow and agencies first deploying a standard may pay a penalty as unforeseen issues in the standard are worked out during the process. Different manufacturers implementing the API Standard independently can also result in operational differences between the implementations which may impair application software interchangeability. At the time of this writing, an opportunity exists with the API Standard development to mitigate these issues and jumpstart deployments.

As a byproduct of the development of the API Standard, portions of the API software were developed by a contractor on the project. The API WG believes it would be highly advantageous to the transportation industry to complete this software as an open source reference implementation of the API Standard. It is proposed that a project be started in spring of 2007 so that the software can be available when the API Standard is scheduled to be published in summer of 2007. Some of the benefits of this open source reference implementation are as follows:

- It facilitates application software portability by providing a common API implementation (diminishes inconsistent operational characteristics due to different software architectures);

- It promotes fast deployment by providing an API implementation in the same time frame as the scheduled publishing date of the API Standard;
- It greatly increases testability of any API implementation for both users and developers;
- It makes the creation of a validation suite relatively "easy;"
- It promotes collaboration of industry software developers;
- It provides a forum for users to express ideas and concerns;
- It promotes quick bug fixes and alternative solutions to issues;
- It facilitates the introduction of new application developers;
- It lowers point of sale and maintenance costs to the industry; and
- It is consistent with the open source concept of the Linux OS in the ATC Controller Standard.

Funding is currently being sought from agencies and manufacturers to support this development effort. It is anticipated that there would be relatively small costs to maintain this software over time with strong industry incentives that would draw volunteer support to keep it up to date. If the reader is interested in supporting this effort or if there are further questions, please contact the author at http://www.pillarinc.com/contact.html.

CONCLUSION

As described in the opening paragraphs of this paper, the transportation industry faces a huge technology gap. There has never been a greater need for advanced technologies in our on-street equipment, yet we continue to deploy new equipment that is underpowered and can do little more than basic traffic control. This means that as the need for more capability is realized, the solution is usually a series of box-level improvements to the traffic cabinet that can never keep up with the demand. Stakeholders of new systems must stop this practice and look towards more flexible and powerful solutions. Systems and equipment need to be selected based on their overall capability. Accordingly, the development of the ATC family of standards means that users have new choices to meet this challenge. The ATC Controller Standard and ATC API Standard work together to provide a powerful general purpose on-street computing platform that is designed to grow with technology.

REFERENCES

1. "ATC Standard Specification for the Type 2070 Controller v2.03", ATC JC, March 12, 2004. Status: User Comment Draft. Update being prepared.
2. "Intelligent Transportation System (ITS) Standard Specification for Roadside Cabinets v01.02.17b", ATC JC, November 16, 2006. Status: Working Group Draft.
3. "Advanced Transportation Controller v5.2b". ATC JC, June 26, 2006. Status: Approved Standard.
4. "ATC Application Programming Interface Standard v02.02". ATC WG, December 3, 2006. Status: User Comment Draft. Approval expected summer 2007.

AUTHOR INFORMATION

RALPH W. BOAZ is President of Pillar Consulting, Inc. where he provides consulting services to the transportation community. He has over 25 years of experience with a broad project management, systems engineering, and software development background. He is a consultant to both the ATC and NTCIP Standards Committees with primary responsibility as the project manager for the ATC API Standard development effort. He also serves as an instructor for ITE's Standards Outreach, Education and Training program and is an invited conference speaker. He holds a bachelor's degree in mathematics from the University of La Verne. He is a member of ITE.

DOUGLAS F. TARICO is a Consulting Software Engineer at Siemens ITS, where he specializes in the development of embedded systems software for transportation management. He has 15 years of experience in the transportation industry, including design and development of real-time control and communication systems, modeling and analysis, and the development of ITS standards. He has been a long-term participant in the ATC standards development effort, currently serving as a member of the ATC Controller working group and as a Co-Chairperson of the ATC API working group. He holds bachelor's and master's degrees in systems engineering from the University of Arizona.