CT-VIEW Introduction

by Michael Manning
CT-View Architect

Visualization in Embedded Development

CT-View offers a unique way to observe the behavior and performance of an application running on an embedded platform. It works in conjunction with your debugger and any Kernel Awareness Tools you may have, or stand alone, without a debugger, Kernel Awareness, or hardware present in some cases.

During a debug session, CT-View can capture System calls while you're running, hitting breakpoints, or single stepping through an application. The tool allows you to highlight and view information such as when a service call occurred, how long it took to execute, and the number of times each service was called. The Track View display shows service call sequence and task behavior, thus providing the developer with a visual perspective and context about the system as it executes.

CT-View may be used during the RTOS evaluation stage of a project, as an informative tool during application development, and within field deployment. Once data is captured, there is no need for the target hardware or a debugger to be present to view and analyze the data, so anyone with CT-View can analyze it.

CT Translation Engine

With the sophistication of today's embedded applications and market pressure to save on design time, the use of abstraction for maintaining portability and re-use of legacy code is a growing prerequisite in product design.

CT-View provides by example, an application that is using an OS agnostic abstraction layer or Translation Engine. The base structure of the Translation Engine (today) contains 56 Core Kernel functions, and 60 Informational Functions totaling 116 services commonly found within today's commercial OS offerings.

The Translation Engine is primarily responsible for mapping CT Services onto their corresponding OS Services by providing a consistent API regardless of which OS is in use. This method of abstraction decouples a CT-View enabled application from the OS making the application portable. It also allows for non-intrusive performance measuring of the underlying OS by wrapping the actual OS calls with time stamp capturing macros for a more accurate representation of the time it takes for a service to complete.

OS Complexity in Variation

Many of the simplest operating system offerings may be lacking in services desirable for a particular application, while at the other end of the spectrum the most sophisticated OS will present complexities in obscure APIs that either require several steps to implement a service, or confuse the developer with non-standard nomenclature regarding use of the service.

In either case there is additional required effort for the developer to either supply a missing service, or learn and implement a more obscure API. The CT-Translation Engine minimizes these issues by providing services that are natively missing, and simplifies the interface to the services contained within the more sophisticated OS structures.

Abstraction – The Performance Trade-Off

With any method of abstraction there is always, at some level, a slight decrease in real-time performance due to the additional code that is running to carry out a specific task. The inherent benefits of abstraction in the majority of embedded applications greatly outweigh the minimal detriment to performance. We recognize that approximately less than 5% of the industry will require the absolute fastest and most efficient use of resources.

The Translation Engine provides a rich API that is easy to use and understand with full coverage of any service an application would need. It supplies a full set of services for Tasks, Queues, Events, Mailboxes, Pipes, Signals, Semaphores, Mutexes, Timers, and HPSRs.

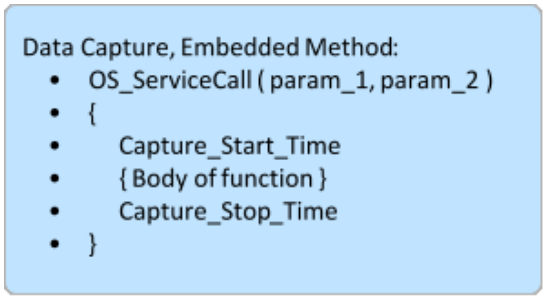

Figure 1

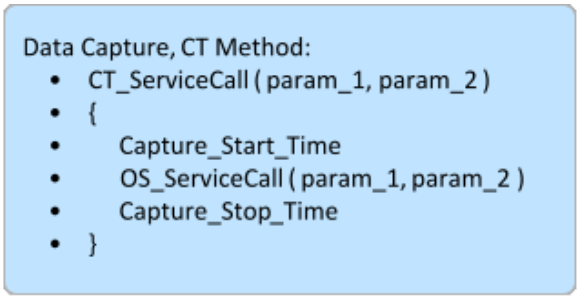

Figure 2

A typical method for capturing performance data is to embed the data capture macros within the body of a function, (Figure 1). While this is an acceptable practice, it does not provide complete accuracy because the time it takes to execute the function's prologue and epilog code is missing. The miniscule amount of time consumed relative to the body of the function may seem insignificant; however the relevance increases over time dependant on the number of times the service is called within a given session.

The CT Data Capture method is more accurate because it includes the prologue and epilog code in the measurements, (Figure2).

The Translation Engine is scalable in that it allows the test suite to be omitted or included at compile time. If the test suite is included at compile time the user may enable or disable it during run-time as needed or at pre-defined time intervals allowing greater flexibility in deployment options with continued testing and diagnostic capabilities.

Further flexibility is inherent with this method allowing for simplicity in measuring a block of code. An example of this would be to measure the time it takes to execute for and while loops allowing the developer to tune them for performance.

The performance macros may also be used in non-CT-View enabled applications and middleware to provide extended or stand-alone data captures within the same visual representation.

Visual Analysis Tool

The CT-Translation Engine is paired with an application which resides on the developer's workstation and transforms the (live) or previously recorded performance data into a visual representation of the activity on the embedded target platform.

Track View – A Merging of Two Seemingly Unrelated Technologies

CT-View has merged the visual capability of a digital multi-track audio recording workstation with the concept of embedded execution flow.

Just as an Audio engineer can “see” clicks, pops and other recording defects – even if they are not noticeable to the untrained ear, CT-View allows the developer to use visual clues and pattern recognition to draw attention to problem areas and speed up the process of correcting problems and optimizing the application.

CT-View visually displays when and where each service or function is called and the relation to everything around it, thus granting the ability to see performance anomalies and how well the sequence of execution parallels toward the developer’s expectation. With multiple zoom levels the developer can drill down and have a closer look even to a nano-second in time scale. This granular detail will help to determine and isolate the specific function or service causing the problem allowing for corrective action and tuning until the desired behavior is obtained.

Inspector View

The Inspector View displays a visual representation of performance data for each service, function call, or block of code that is time-stamped within the embedded application. The data is represented by bar graphs that provide fastest, slowest and average execution times for the specified service during a live capture or session playback.

This feature is particularly useful for side by side comparisons, such as the performance impact of using different Real-Time Operating Systems.

The visual aid of the Inspector View also provides a developer with the perspective of possible anomalies and bottlenecks. High numbers, or nearly filled graphs, would indicate that a particular service, on average is taking a long time to complete, providing an immediate identifier subject of further investigation. Visual recognition of skewed or unexpected execution times helps the developer understand where to look within the given time line to quickly identify problems such as an un-wanted context switch or perhaps synchronization issues with other components.

In any case, simple visual feedback leads to faster editing and fine tuning which ultimately produces efficient and predictable results.

Features

CT-View contains various features that assist with presenting the developer with useful information about the activity that is occurring on the target.

Live Capture
The developer can simply run the application on the target and view activity while the capture or session is in progress.

Session Playback
Every session is automatically saved to a file named "temp_log.ctv" on the developer's hard disk in the folder where CT-View is installed. Independent captures may be loaded and played back as often as desired. Log files may be renamed and stored for comparison to future sessions.

Loop Playback
The developer can isolate any section of activity and repeatedly loop through the defined beginning and end point during playback or while single stepping though each event.

Zoom
CT-View provides 15 levels of zoom to allow developers to view granular activity down to a nano-second scale.

Execution Sequence
CT-View provides a trace marker that may be toggled on/off and is used to highlight the sequence of execution flow.

Alarms
Adjustable alarms allow the developer to specify execution time thresholds in microseconds to trigger display markers. Toggling the Alarms view on/off will automatically change the track view markers to execution time bars.

Track View - Control
Solo & Omit selections are available to clean and isolate different viewing perspectives.

Color Themes
Allows for the selection of pre-defined Color Themes (User Preference)

Navigation Bar

- Loop Playback
- Jump to Start
- Play Slower
- Stop Playback
- Playback/Animate
- Step
- Capture/Record
- Play Faster
- Jump to End

Color Coding
Each Event type is displayed in a separate color to assist the developer with the ability to recognize what service is being accessed. Detailed information can be obtained by using a cursor hover over the event.

Code Block Coverage
In addition to OS Services, CT-View will display markers and performance data for the user's application functions or code blocks with no additional requirement from the developer other than applying the time stamp macros to the desired capture points within the application.

Custom Visibility
Performance macros (provided), may be inserted anywhere within user application code providing visibility into the behavior and performance of any function or section of code the developer may require. The performance data for each capture point will be displayed in the Inspector View, and the behavior characteristics will be displayed in real time within the Track View.