HighIntegritySystems

WITTENSTEIN

Saving Power Through Software Development

Issue 1.1 - 01 May, 2024

WITTENSTEIN high integrity systems
Americas: +1 408 625 4712
ROTW: +44 1275 395 600

email: sales@high**integrity**systems.com
web: www.high**integrity**systems.com

WITTENSTEIN

CONTENTS

List of Figures

List of Notation

ADAS	Advanced Driver Assist System
CPU	Central Processing Unit
GWh	Gigawatt hour
J	Joules
kWh	Kilowatt Hours
mA	Milliamps
MCU	Microcontroller Unit
mW	Milliwatts
MWh	Megawatt Hours
RTOS	Real Time Operating System
SoC	System on Chip
USARTs	Universal Synchronous and Asynchronous Receiver-Transmitter
V	Volts
W	Watts
WFI	Wait for Interrupt
Wh	Watt Hours
Ws	Watt Seconds

CHAPTER 1 Introduction

1.1 How Does Software Impact The Climate?

The carbon footprint attributable to software is rapidly growing. The majority of organizations acknowledge the need to mitigate greenhouse gas emissions, recognizing carbon dioxide, a by-product of fossil fuel energy sources, as a key contributor to the greenhouse effect. “Net zero carbon” goals are now widespread. Nevertheless, as the transition to electric vehicles for transportation and heat pump technology for residential heating gains momentum, the demand for electricity is poised to surpass our current renewable and nuclear capacity. Consequently, this surge in demand translates into a corresponding increase in carbon dioxide emissions in the foreseeable future.

An important driver of electricity demand stems from the rapid proliferation of programmable devices and their accompanying infrastructure. Although these devices are typically more efficient than their predecessors, their sheer abundance translates into staggering energy requirements. Recent estimates indicate that the power consumption of digital devices and networks is escalating at more than twice the rate of the overall increase in electricity demand.

The power consumption of digital devices is not solely determined by hardware; software plays a crucial role in influencing energy usage. As conscientious engineers, we can strive to maximise the value derived from each gram of carbon dioxide emitted. It behoves us to design both hardware and software to minimize environmental impact.

1.2 What Does This Paper Cover?

This paper will delve into power-saving strategies for embedded software applications employing an embedded RTOS. Initially, we will explore a proof-of-concept developed by WITTENSTEIN high integrity systems (WHIS), illustrating how a single line of code integrated into the idle task of an RTOS application can save over a third of the total energy consumed by the processor. Subsequently, we will discuss comprehensive power-saving measures across the entire embedded system, focusing on the utilization of energy-efficient techniques. Following this, we will briefly explore specific energy-saving methodologies tailored for automotive and industrial software applications.

Engineers have a pivotal role in shaping the power consumption of future devices. Given the immense task of constructing carbon-efficient applications, where should one begin? It starts with recognizing that the design, development, and deployment of software wield significant influence over energy consumption.

CHAPTER 2 How To Save Power With Your RTOS

2.1 Saving Power Through Coding: The Full Case Study

Many common processors have one or more specific power saving instructions that put the CPU into a low power mode until an interrupt occurs. On Arm Cortex-M processors there is an instruction called WFI – Wait for Interrupt. WFI enters low power mode but leaves the system tick timer running at normal speed. To effect power savings on the ARM Cortex-M processors, or any processor with a similar instruction available, is simple when using a pre-emptive priority based RTOS. All that is needed is to place a single WFI instruction in the RTOS idle hook function, which is typically the lowest priority task and is only executed when the RTOS has no other task to schedule. As soon as the idle task runs, it calls the idle hook, hits WFI and the processor goes into a low power sleep until the next interrupt occurs. This is usually the tick interrupt for the next system tick. Full power operation is resumed until the next time the idle hook function is called.

The implementation on an ARM Cortex-M platform is simple:

```
void vApplicationIdleHook( void )
{
     asm volatile (“ WFI \n” );
}
```

2.2 Proof of Concept

WHIS used the simple LED colour cycling application from the Workshop Demo “Upgrading from FreeRTOS to SAFE**RTOS**®” on an off the shelf Cortex-M processor. It has several tasks that are busy but all of them spend most of their time blocked, so the processor spends a large proportion of every tick period in the idle task. We therefore created an idle hook function exactly as above with a single WFI in it. To quantify the amount of power we could save, we ran it with the WFI in place and then compared it to the same application with the WFI instruction commented out.

To compare power consumption, as shown in figure 1, we modified the board on which the demo runs, breaking a solder bridge in the supply rail to the target CPU and inserting a jumper link that could be replaced with a digital multimeter reading current in milliamps (mA). Power (in Watts) is current (in Amps) times voltage (in Volts), and energy consumption is power (Watts) times time – either Watt Seconds for energy in Joules (J) or typically Watt Hours (W.h) for domestic electricity billing. Current was of the order of 50 mA, and the processor supply is 3.3v, which is approximately 150 mW.

	FreeRTOS base task	**FreeRTOS Task with WFI instruction**	SAFE**RTOS**® **base task**	SAFE**RTOS**® **Task with solution**
Power used	150.5 mW	90.3 mW	145.0 mW	90.4 mW
Energy per year	1.32 kWh	0.79 kWh	1.27 kWh	0.79 kWh
Savings/year in 1 deployment	-	0.53 kWh	-	0.48 kWh
Savings/year in 50,000 deployments	-	26.37 MWh	-	23.91 MWh
Savings over 5 years in 1,000,000 deployments	-	2.64 GWh	-	2.39 GWh

Figure 1 *Power Consumption*

Note: The multimeter will have introduced some resistance in order to measure current and that resistance will have reduced the core voltage from the nominal 3.3 volts. The core will still use the same power, so the current will be higher than it would be without the meter in circuit, which, when multiplied by the nominal 3.3 volts gives a power consumption figure slightly higher than actual. The difference is small, though, and all readings are affected the same way, so the comparisons with/without WFI in idle hook are valid. Application differences will be vastly more significant in actual practice.

2.3 Power Saving Explained

2.3.1 Without Power Saving

In the usual case, with no power saving measures, the processor is always busy and using around the same amount of power. We can illustrate a typical pattern of tasks, interrupts and idle time as shown in figure 2.

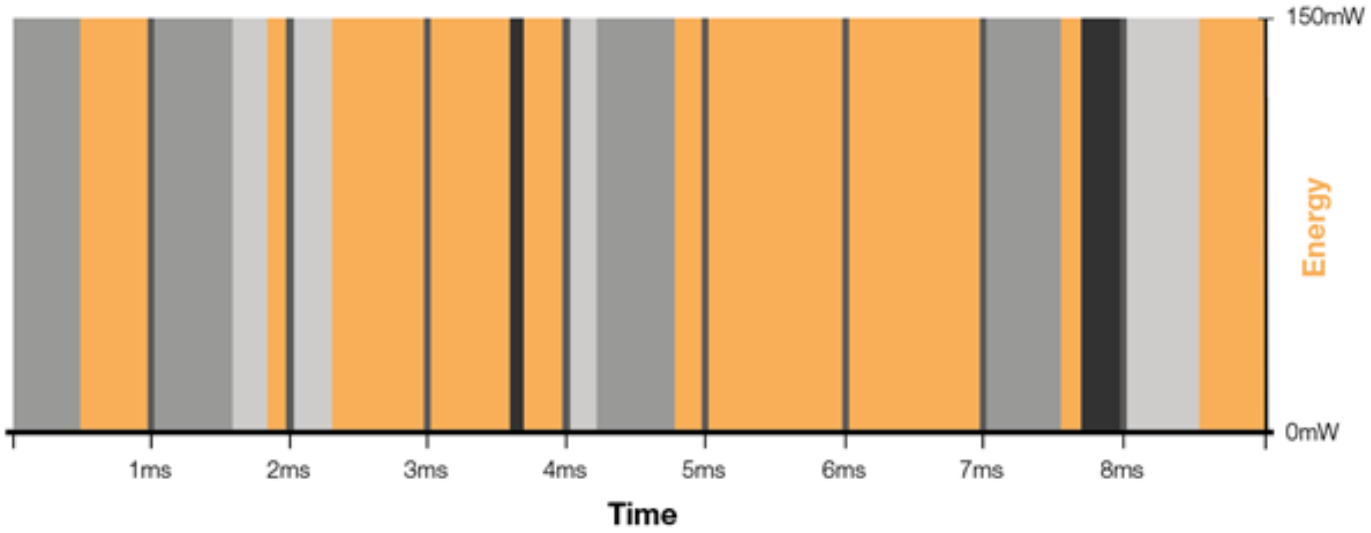

Figure 2 *Usual Case*

Key: Represents time spent in idle task Represents the system tick interrupt at (say) 1ms intervals Represents a random interrupt Both represent task activities.

When the SAFE**RTOS**® scheduler has nothing else to do, it selects the idle task as the processor does not stop. This simply sits in a loop either doing nothing, or repeatedly calling an "idle hook function" if the application writer has provided one. FreeRTOS behaves in the same way. The idle hook function might be used to do various background tasks that are non-urgent, or it might be used to make some attempt to save power.

2.3.2 Power Saving using the Idle Hook

If we place a WFI in the idle hook, as soon as we hit the idle task, power use is cut down until the next interrupt occurs. This can be demonstrated in figure 3.

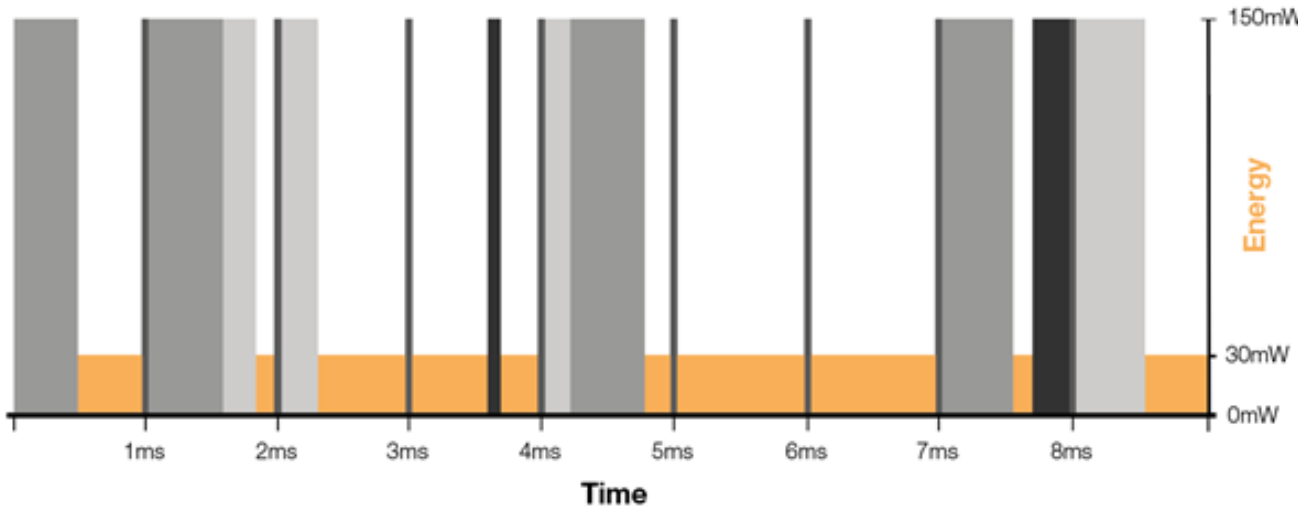

Figure 3 *With WFI In The Idle Hook*

Key: Represents time spent in idle task Represents the system tick interrupt at (say) 1ms intervals Represents a random interrupt Both represent task activities.

Now, whenever it is in the idle task, the processor is halted or slowed and power consumption is significantly reduced. Despite this, the tick timer still runs and external interrupts can be processed. For each interrupt, the processor wakes up and returns to full power to handle the interrupt and can resume a task if one is ready to run. If the only task that can run is the idle task, it returns to that, which calls WFI again.

2.3.3 "Tickless" port

In a Tickless port, whenever the processor enters the idle task, the kernel code determines when the next task is scheduled to run. If that is more than the pre-defined number of tick threshold value (typically one) then the kernel programs an external timer to wake it up in that many ticks in the future. The kernel turns off the tick interrupt timer and other functions for greater power savings. The processor will then be woken by the pre-programmed future tick (and usually by any other interrupts that happen to occur in the meantime). When it wakes on the future tick, the internal tick counter is advanced as if the intermediate ticks had been processed. This can be seen in figure 4.

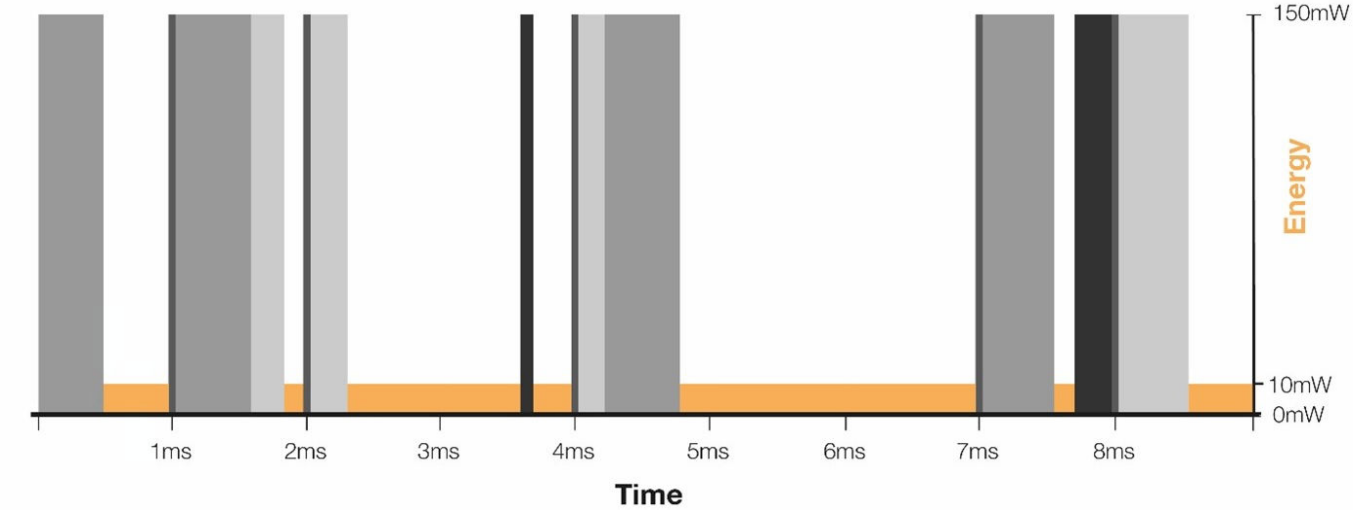

Figure 4 *Tickless Port*

Key: Represents time spent in idle task Represents the system tick interrupt at (say) 1ms intervals Represents a random interrupt Both represent task activities.

As shown in figure 4, the ticks at 3ms, 5ms and 6ms have not occurred because they were not needed to wake any task and the external timer has been used instead to wake the processor at 4ms and 7ms. The power consumption in the idle, after the call to WFI, is even lower because the processor peripheral busses that drive parts similar to its internal tick timer etc. have also been switched off. It is apparent that the optimal approach, and the overall amount of power saving possible, is dependent on the application's pattern of idle task usage. Moreover, by subtly altering the way in which the application behaves, power use can be potentially dramatically affected.

CHAPTER 3 Power Saving Through The Entire Embedded System

When designing an embedded system with power saving in mind, the entire design must be considered.

3.1 Choice of Processor

Selecting a processor that aligns precisely with your application requirements without exceeding them is paramount. Overpowered processors consume unnecessary energy if their capabilities aren't fully utilized by your application. Many processors incorporate power-saving functionalities, albeit with variations across devices. Opting for a processor equipped with such features can significantly reduce energy consumption. However, beyond these considerations, it's essential to evaluate the processor's wake-up speed.

The choice of processor often involves navigating through numerous, sometimes conflicting, requirements. Fortunately, most manufacturers offer lower-power versions of popular MCUs, with newer iterations typically boasting enhanced power-saving capabilities. When contemplating a new product version poised for widespread deployment, it may be advantageous to transition to a different architecture if it promises substantial efficiency gains. Nonetheless, it's important to note that alternative variants often come with additional power-saving attributes such as deep sleep modes, optimized wake-from-sleep performance, streamlined feature sets, or enhanced component integration, which may necessitate additional devices.

3.2 Choice of Peripherals

Within the processor, when designing the application, it is important to turn off the peripherals that are not needed in the final product. This will create a lower energy output for the total system, as well as easier integration and greater reliability. On the whole, designs with a higher integration of functions are more reliable, as the IO ports are being driven in a power efficient manner and less of the working circuit is exposed to external interaction, such as driving output signals etc. Often processors will have subsystems that a design does not use.

Optimizing efficiency entails selecting a processor variant tailored to your design's specific needs, avoiding unnecessary features that remain dormant Leveraging available processor facilities, such as the ability to selectively power down components like USARTs except during intermittent communication, can yield significant energy savings. Generally, highly integrated system-on-chip (SoC) solutions tend to be more efficient than systems relying on multiple standalone devices. However, if an SoC incorporates rarely used, large subsystems that cannot be fully powered down, integrating an external system with the capability to disconnect power may offer superior efficiency. Conversely, certain subsystems may operate independently, enabling the processor itself to enter extended periods of shutdown, further enhancing efficiency.

3.2 Choice of Operating Frequency

In simple terms, the faster a processor operates, the greater its electricity consumption. Opting for a slower chip speed can be a valuable power-saving feature if feasible. However, selecting the operating frequency isn't a straightforward decision. Generally, higher frequency operation consumes more power, but there's a delicate balance between maximizing idle time in low-power modes and minimizing power usage during active operation. Determining the optimal trade-off depends on specific application requirements and design considerations. If an application necessitates frequent CPU activity but experiences low overall system load, reducing the operating frequency can conserve power. Conversely, if the processor is only intermittently active, employing a relatively high frequency may enhance power savings by minimizing the duration of processor activity. This determination hinges on the precise workload and power consumption at different frequencies.

For optimal power efficiency, a design may require dynamically varying CPU operating frequencies. Given the complexity and variability of applications, there's no one-size-fits-all approach. Achieving truly optimal power consumption necessitates a thorough analysis and adjustment of every design aspect with power-saving objectives in mind.

3.2 Application Design

With escalating consumer demand for environmentally friendly, energy-efficient products, engineers face the imperative task of minimizing the power consumption of their designs. Considering that the majority of electricity is generated through the burning of fossil fuels, energy supply stands as the primary contributor to carbon emissions. Enhancing the efficiency of your application is one of the most effective methods for reducing electricity consumption.

WHIS has already demonstrated a notable approach in this regard. Typically, even when idle, applications maintain a static power draw, necessitating the implementation of low-power modes in many processors. WHIS' findings reveal that integrating a Wait For Interrupt Instruction (WFI) into the idle task of a Cortex-M processor can yield energy savings of up to one-third. By embedding power-saving mechanisms into your architectural design, energy conservation becomes more attainable.

Employing an RTOS within a task-oriented design offers a strategic advantage. Tasks are prioritized based on their level of importance, and during periods of inactivity, the application seamlessly transitions to the idle task, an ideal opportunity for energy-saving measures. While the Idle Task may perform background processing, the system predominantly remains in a state of readiness until an event prompts the activation of a task. In contrast, non-RTOS application designs often rely on continuous sequence calls, leading to a perpetual execution loop.

Furthermore, an RTOS enhances code cleanliness and organization, providing a structured framework for feature development. Thus, even in scenarios where real-time capability is unnecessary, an RTOS remains a favorable option.

Silicon devices and microprocessors compatible with an RTOS offer a plethora of low-power modes. These modes enable selective shutdown of various processor components, such as peripheral units or unused oscillators, when not in use. While wake-up times from low-power modes vary depending on the microprocessor and its configuration, they serve as invaluable energy-saving functions. An RTOS streamlines the transition to low-power modes by prioritizing tasks and scheduling them at intervals, simplifying the process and facilitating efficient energy management.

CHAPTER 4 Power Saving Across Industries

4.1 Automotive

Cars are an easy target for deploying a huge number of embedded processors. A modern car can easily have more than 3,000 chips with 100-plus microprocessors [1]. With advanced driver-assistance systems (ADAS) and autonomous vehicles on the horizon and software safety criticality becoming more important daily, software will continue to have more control over the vehicle. As the landscape shifts, automakers will have to utilize their software like never before, as the complexity of automotive software is escalating on both the functional and architectural levels [2]. Reducing architecture complexity and software design for the automotive industry is not only important to help increase power saving but also to win the software game.

Safety is critical in the automotive sector, where timely deployment of airbags and anti-lock braking systems can mean the difference between life and death. Even the slightest timing error could have catastrophic consequences. This underscores the neccesity for a RTOS, as it provides assurance to the driver and the passengers of the car as well as the application designer that all operations will act according to their time restraints. In an automotive vehicle, multiple software tasks are running on individual microprocessors. An RTOS system has a high degree of control over the prioritization of tasks. By leveraging this heightened level of control, the system can be designed more efficiently.

4.2 Industrial Devices

During the past decade, industrial applications have seen an ongoing increase in the adoption of digital technologies; it has been called the new industrial revolution [3]. To keep pace with the market's demands, highly efficient and safe technologies are needed to support industrial evolution. When looking for embedded designs for industrial devices, safety and power are key with technology adoption at the heart of success. With customers and suppliers expecting more than ever, it is reasonable to expect that industries will seek to be more efficient. This will include embracing new technologies such as digitization, artificial intelligence and robotics. From this, industries stand to drive sustainable growth [4].

With new automation comes more pressure on companies to reduce their energy emissions, as more companies are focusing on sustainability not only for their business but also for the planet. As with the automotive sector, software will start to have more control over industrial devices and system architectures with software design will become vital in the wider context of success and sustainability. By specifically supporting the needs of industrial device developers, a safety RTOS can greatly reduce program risks, lower development costs and shorten the time to market for industrial device products. Not only this, but as this white paper has shown, it is possible to save significant power on your application.

Safety is paramount in industrial settings, where stringent requirements govern the operation of machinery and the mitigation of technological risks is imperative. Considerations such as heat management in deeply embedded or physically constrained environments, as well as precise process control in motor operations, underscore the need for meticulous attention to detail. Moreover, in the realm of mass production, striking a balance between power conservation and maintaining quality is essential, particularly in battery-operated applications like electric vehicles.

CHAPTER 5 Conclusion

This white paper has looked at what saving power on software typically means in an embedded system. It is evident that we need transformational changes to address the climate crisis - a transformation that will rely on technology, innovation, and cross-industry collaboration.

We have showcased the substantial scalability of power savings achieved by incorporating just a single line of code across multiple processors. At first glance, saving a small amount of power for a single application running on one processor might appear inconsequential. It is tempting to overlook optimization efforts for such seemingly straightforward applications. However, when we contemplate the vast scale of electronics deployed across industries, even the minutest adjustments assume significant importance.

In the pursuit of reducing carbon consumption and fostering sustainability, application designers must prioritize minimizing energy consumption throughout the entire lifecycle of their products. To do this, it is essential that energy considerations are incorporated within the design phase of a product, since the majority of a product's environmental impact is determined during this phase.

Embedded safety software is an exciting field to be a part of, especially at this time of exponential growth. WITTENSTEIN high integrity systems is glad to be able to share this information with you, in a hope that engineers will have another tool to save energy wherever possible.

References

[1] The New York Times, A Tiny Part's Big Ripple: Global Chip Shortage Hobbles the Auto Industry [online] https://www.nytimes.com/2021/04/23/business/auto-semiconductors-general-motors-mercedes.html#:~:text=One%20big%20reason%20automakers%20can,have%20more%20than%203%2C000%20chips; Forbes, Chip Shortages Force Automakers To Slash Production [online] https://www.forbes.com/wheels/news/chip-shortages-force-automakers-slash-production/

[2] McKinsey & Company, When code is king: Mastering automotive software excellence [online] https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/when-code-is-king-mastering-automotive-software-excellence

[3] McKinsey & Company, Facing the future: Britain's new industrial revolution [online] https://www.mckinsey.com/business-functions/operations/our-insights/facing-the-future-britains-new-industrial-revolution

[4] K. Schwab, "The Fourth Industrial Revolution: what it means, how to respond," World Economic Forum, January 14, 2016, weforum.org.

CONTACT INFORMATION

User feedback is essential to the continued maintenance and development of SAFE**RTOS**®. Please provide all software and documentation comments and suggestions to the most convenient contact point listed below.

Contact WITTENSTEIN high integrity systems

Address: WITTENSTEIN high integrity systems
Brown's Court, Long Ashton Business Park
Yanley Lane, Long Ashton
Bristol, BS41 9LB
England

Phone: +44 (0)1275 395 600
Email: support@HighIntegritySystems.com

Website www.HighIntegritySystems.com