



circuit cellar

Inspiring the Evolution of Embedded Design

RAILWAY SYSTEMS TRAVERSE EMBEDDED COMPUTING TRACK



■ Product Focus: IoT Interface Modules ■ FPGA Signal Processing |

Inrush Current Limiters | Macros for AVR Assembler Programming |

Smart Frying Pan | Eye-Controlled Video Game ■ Cores with TrustZone-M |

Energy Monitoring (Part 2) | Variable Frequency Drive (Part 1) |

Windless Wind Chimes (Part 2) ■ The Future of Autonomous Cars



You can take it almost anywhere.
**Where will it
take you?**



CC VULT

Now you can have the complete Circuit Cellar issue archive and article code stored on a stylish, durable and portable USB flash drive. You can easily keep your CC Vault archive up to date by purchasing subsequent issues from our webshop or by downloading issues with a Circuit Cellar Digital Subscription. Issues appear in searchable PDF format.

Complete archive includes PDFs of all issues in print through date of purchase.

Visit cc-webshop.com to purchase





The Embedded Experts



emCompress-ToGo

Compress Data in Real-time on any Embedded System!



More Storage



More Bandwidth



Faster Updates

One Professional Compression Solution for All Applications



Data Loggers



Internet of Things



Space / Avionics



Networking



Medical Devices



Consumer Electronics

- Real-time compression
- Small footprint
- No static RAM required
- Compression of data stream
- High performance
- High compression ratio
- On-target compression & decompression

Worldwide: sales@segger.com

+49 2173 99312 0

U.S. East Coast: us-east@segger.com

+1 978 874 0299

U.S. West Coast: us-west@segger.com

+1 408 767 4068

segger.com

OUR NETWORK



SUPPORTING COMPANIES

Accutrace, Inc.	C3
Aceinna	25
All Electronics Corp.	77
CCS, Inc.	77
Hackerboxes	31
Hot Chips Symposium	63
Labcenter Electronics, Ltd.	17
SEgger Microcontroller Systems	1
Slingshot Assembly	21
Technologic Systems, Inc.	C4, 77
University of Cincinnati	11
UrsaLeo, Inc.	35
VersaLogic Corp.	47

NOT A SUPPORTING COMPANY YET?

Contact Hugh Heinsohn
(hugh@circuitcellar.com, Phone: 757-525-3677, Fax: 888-980-1303)
to reserve space in the next issue of *Circuit Cellar*.

THE TEAM

PRESIDENT
KC Prescott

EDITOR-IN-CHIEF
Jeff Child

ADVERTISING COORDINATOR
Nathaniel Black

CONTROLLER
Chuck Fellows

SENIOR ASSOCIATE EDITOR
Shannon Becker

ADVERTISING SALES REP.
Hugh Heinsohn

FOUNDER
Steve Ciarcia

TECHNICAL COPY EDITOR
Carol Bower

PROJECT EDITORS
Chris Coulston
Ken Davidson
David Tweed

GRAPHICS
Grace Chen
Heather Rennae

COLUMNISTS

Jeff Bachiochi (From the Bench), Bob Japenga (Embedded in Thin Slices), Robert Lacoste (The Darker Side), Brian Millier (Picking Up Mixed Signals), George Novacek (The Consummate Engineer), and Colin O'Flynn (Embedded Systems Essentials)

Issue 348 July 2019 | ISSN 1528-0608

CIRCUIT CELLAR® (ISSN 1528-0608) is published monthly by:

KCK Media Corp.
PO Box 417, Chase City, VA 23924

Periodical rates paid at Chase City, VA, and additional offices. One-year (12 issues) subscription rate US and possessions \$50, Canada \$65, Foreign/ ROW \$75. All subscription orders payable in US funds only via Visa, MasterCard, international postal money order, or check drawn on US bank.

SUBSCRIPTION MANAGEMENT

Online Account Management: circuitcellar.com/account
Renew | Change Address/E-mail | Check Status

CUSTOMER SERVICE

E-mail: customerservice@circuitcellar.com

Phone: 434.533.0246

Mail: Circuit Cellar, PO Box 417, Chase City, VA 23924

Postmaster: Send address changes to
Circuit Cellar, PO Box 417, Chase City, VA 23924

NEW SUBSCRIPTIONS

circuitcellar.com/subscription

ADVERTISING

Contact: Hugh Heinsohn

Phone: 757-525-3677

Fax: 888-980-1303

E-mail: hheinsohn@circuitcellar.com
Advertising rates and terms available on request.

NEW PRODUCTS

E-mail: editor@circuitcellar.com

HEAD OFFICE

KCK Media Corp.
PO Box 417
Chase City, VA 23924
Phone: 434-533-0246

COPYRIGHT NOTICE

Entire contents copyright © 2019 by KCK Media Corp. All rights reserved. Circuit Cellar is a registered trademark of KCK Media Corp. Reproduction of this publication in whole or in part without written consent from KCK Media Corp. is prohibited.

DISCLAIMER

KCK Media Corp. makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors printed in Circuit Cellar®. Furthermore, because of possible variation in the quality and condition of materials and workmanship of reader-assembled projects, KCK Media Corp. disclaims any responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in Circuit Cellar®.

The information provided in Circuit Cellar® by KCK Media Corp. is for educational purposes. KCK Media Corp. makes no claims or warrants that readers have a right to build things based upon these ideas under patent or other relevant intellectual property law in their jurisdiction, or that readers have a right to construct or operate any of the devices described herein under the relevant patent or other intellectual property law of the reader's jurisdiction. The reader assumes any risk of infringement liability for constructing or operating such devices.

© KCK Media Corp. 2019 Printed in the United States

INPUT Voltage

There was a clever phrase that we in the bus & board industry used to use a lot in the mid '90s: "Zero Billion Dollar Market." When I say bus & board industry, I'm referring to the community of folks involved in standard embedded computing architectures such as VME, CompactPCI, COM Express and so on. One way to check someone's age in that community is to ask them if they remember Multibus II, STD bus or the ironically short-lived FutureBus. I can't say that I coined the phrase Zero Billion Dollar Market, but I can claim that I was among the circle of folks that first started using it. It's a way to describe a technology market that's in its infancy, while at the same time has huge expectations attributed to it.

If memory serves—no pun intended—I first used Zero Billion Dollar Market to describe a particular type of flash memory that, at that time, in the mid-'90s, was brand new but was expected to quickly become a market with revenues in the \$billions, depending on whether the consumer electronics industry chose to design it in or not. And if the answer turned out to be "not"? Well, the math is pretty simple: \$0 billion = \$0. Game over. In that case, I believe that flash memory architecture did indeed take off and earn its billions.


Around that same time, the phrase Zero Billion Dollar Market started to be used to describe the CompactPCI standard embedded board form factor. To be more specific, it was applied to CompactPCI in terms of expectations that it would enjoy huge uptake by telecom system integrators. CompactPCI never did gain any large scale acceptance in telecommunications. Yet it caught on famously in a variety of markets such as industrial control, instrumentation and even mil/aero—and by any account it can be called a success.

Now let's talk about the Internet-of-Things (IoT). I have to admit that if you asked me what I thought of IoT around 5 or 6 years ago, I was on the skeptical side. And I would have comfortably applied the Zero Billion Dollar market accolade to IoT. I'd seen numerous catch phrases like IoT come and go in the embedded systems industry, and I thought it would be short lived. As IoT gained momentum, it was clear that my skepticism was unfounded. Today, IoT is spread across several market areas with industrial, transportation, smart homes and energy segments growing fastest. Even if you examine IoT in terms on embedded devices using processors,

microcontrollers, wireless connectivity, sensors and high-level operating systems—you're still talking billions just in unit numbers alone.

Many of the latest market research reports that look at various aspects of industrial IoT show a bright future ahead. For example, a report in April by Market Study Report forecasts the global industrial IoT market valuation is to reach \$771.72 billion by the end of 2026. According to the report, much of the growth is being driven by organizations looking to improve overall operational efficiency, productivity and visibility, while also reducing the complications of diverse procedures in the business sphere. Industrial IoT is seen as a way to achieve significant cost reduction. Other factors driving IoT acceptance include declining sensor prices which have led to a reduction of total costs associated with data collection and analytics.

According the analysis by Market Study Report, government-sponsored initiatives are expected to help drive growth as well. For example, Industrie 4.0 is the German government-sponsored multi-year strategic initiative to unite major public and private sector companies along with academia to design a strategy for implementing digital technologies in the country's industrial sector. Meanwhile, in the Asia Pacific market, initiatives by the Chinese government such as Made in China 2025 is supporting the integration and promotion of digital technologies within the country's industrial sector. In terms of regional growth, the Asia Pacific industrial IoT market is estimated to grow at the fastest CAGR due to increasing adoption of IIoT technologies in nations like China, Japan and Taiwan.

There's no doubt that the forecasts for industrial IoT paint a successful future ahead. And while the numbers may vary either way, when it comes to this market, I think I'll have to put my Zero Billion Dollar Market phrase away for now and save it for next "big" thing. 



Jeff Child

COLUMNS

PRODUCT FOCUS

48 IoT Interface Modules Smart Solutions

By Jeff Child

52 Picking Up Mixed Signals Variable Frequency Drive (Part 1) Washing Machine Repurposed

By Brian Millier

58 Embedded System Essentials A Look at Cores with TrustZone-M Security Scrutinized

By Colin O'Flynn

64 The Consummate Engineer Energy Monitoring (Part 2) Tracking Electric Power

By George Novacek

68 From the Bench Windless Wind Chimes (Part 2) My MIDI Upgrade

By Jeff Bachiochi

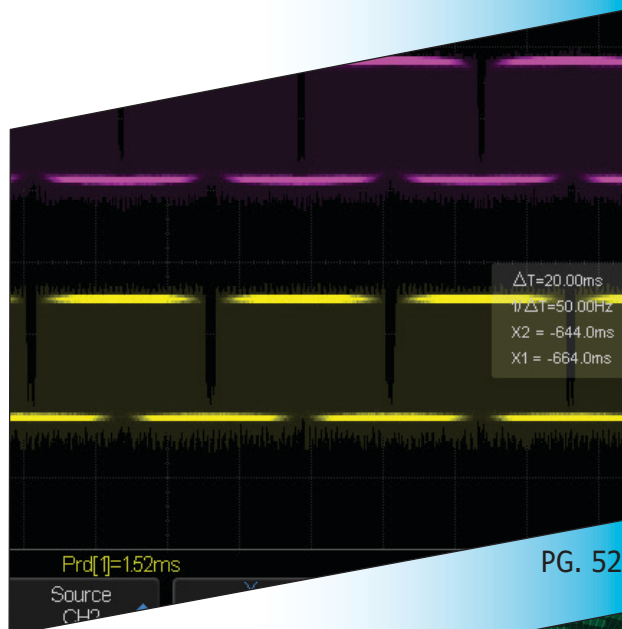
TECH THE FUTURE

79 The Future of Autonomous Cars Sensors, Software and More Sensors

By James Fennelly

76 : PRODUCT NEWS

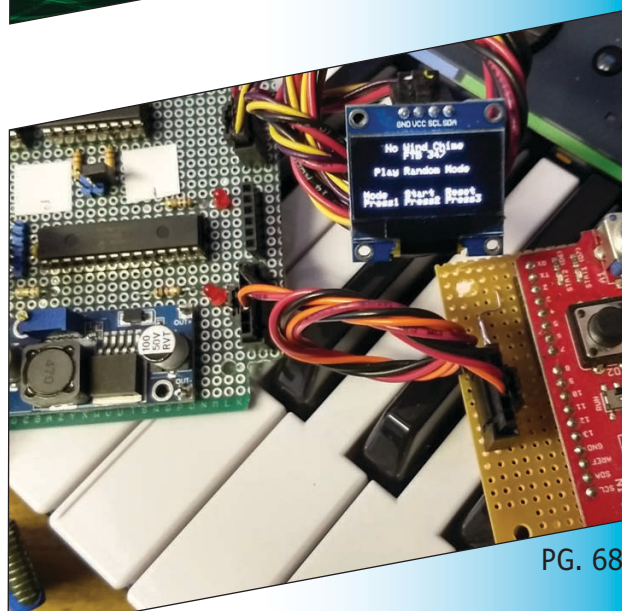
78 : TEST YOUR EQ



PG. 52

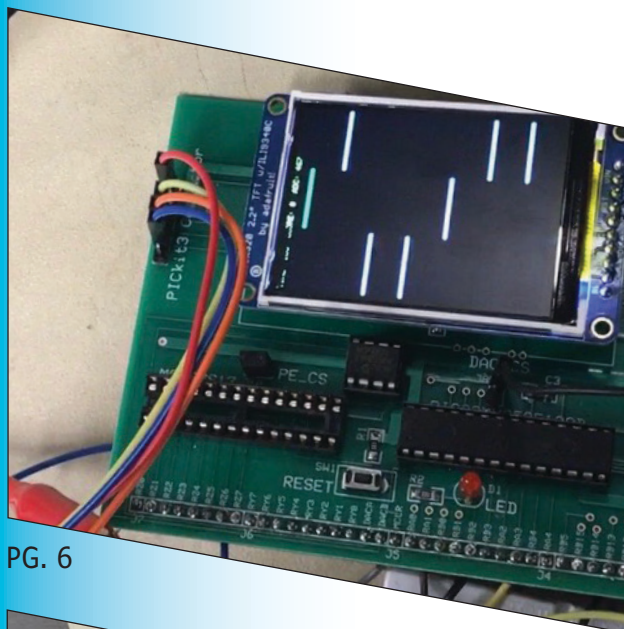


PG. 58

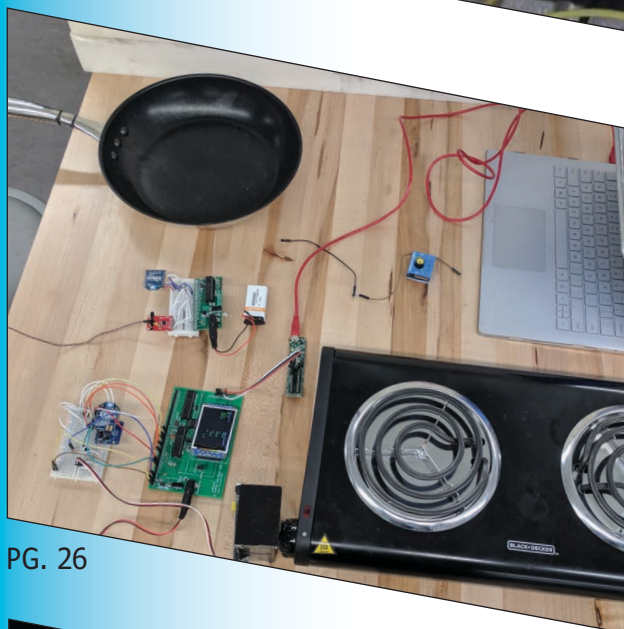


PG. 68

FEATURES



PG. 6



PG. 26



PG. 44

6 EOG-Controlled Video Game Eyes as Interface

By Eric Cole, Evan Mok and Alex Huang

12 Macros for AVR Assembler Programming

*Tools of the Machine Code Trade**By Wolfgang Matthes*

26 Building a Smart Frying Pan Connected Control for Chef

By Joseph Dwyer

32 Inrush Current Limiters in Action

*Circuit Guardians**By Matt Reynolds*

36 Embedded Solutions Enable Smarter Railway Systems

*Computing, Connectivity and Control**By Jeff Child*

44 TECHNOLOGY SPOTLIGHT FPGAs Flex Their DSP Muscles

*Pros at Signal Processing**By Jeff Child*

EOG-Controlled Video Game

Eyes as Interface

There's much to be learned about how electronics can interact with biological signals—not only to record, but also to see how they can be used as inputs for control applications. With ongoing research in fields such as virtual reality and prosthetics, new systems are being developed to interpret different types of signals for practical applications. Learn how these three Cornell graduates use electrooculography (EOG) to control a simple video game by measuring eye movements.

By *Eric Cole, Evan Mok*
and *Alex Huang*

The human eye naturally acts as a dipole, in which the retina at the back of the eye is negatively charged, and the cornea at the front of the eye is positively charged. EOG is a recording technique that measures this potential difference, and can be used to quantify eye movement [1]. A typical electrode placement pattern for EOG is shown in **Figure 1**. Each of the electrodes A and B records a voltage related to eye movement,

and an electrode at point C serves as a ground reference.

When a user looks left, the cornea is close to electrode B and it records a positive voltage, while the retina is closer to electrode A, yielding a negative voltage. Similarly, looking right produces a negative voltage at B and a positive voltage at A. The difference between V_B and V_A relative to ground at C changes monotonically with gaze direction, and can be reliably used to model horizontal eye movement.

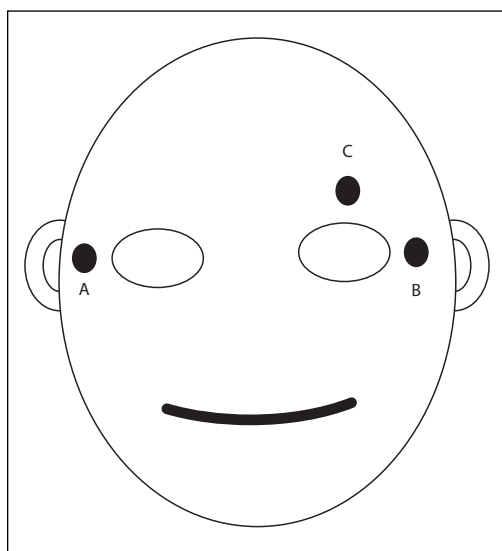
SYSTEM OVERVIEW

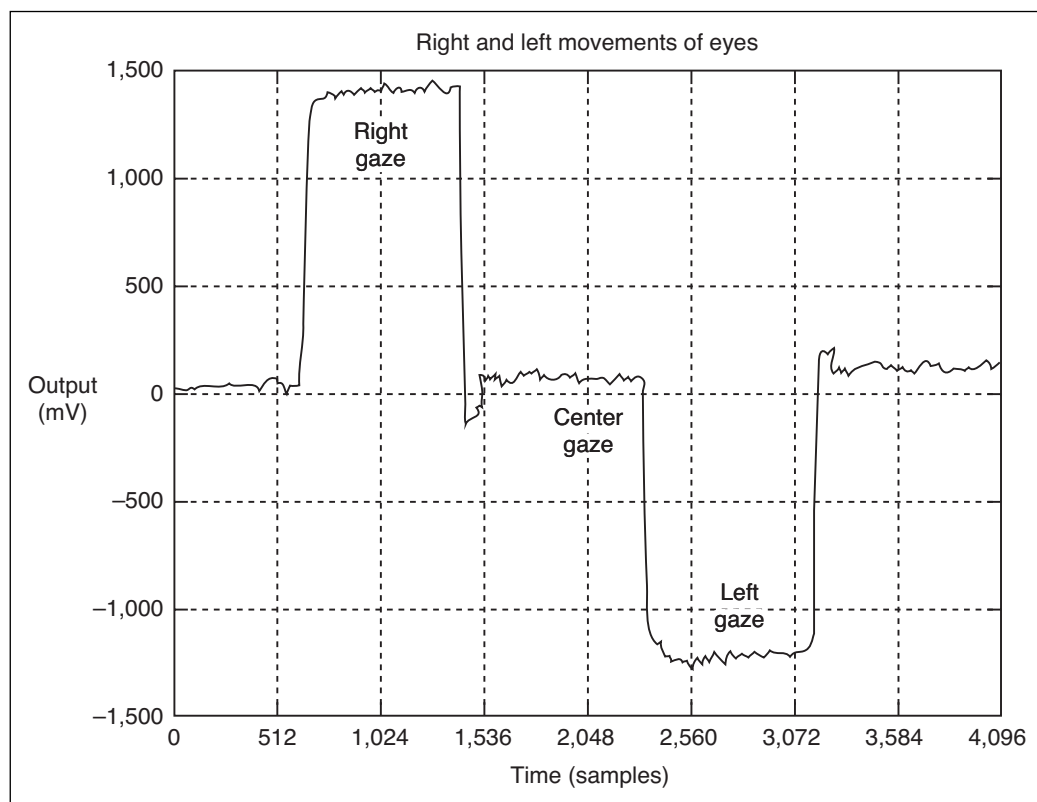
The system we designed uses eye movements to play a video game on a display screen. Electrodes are placed on a player's head to record only the horizontal EOG signal as shown in **Figure 2**. This signal is then filtered and amplified via an analog circuit and sent to an ADC on a Microchip Technology PIC32 microcontroller (MCU) (**Figure 3**). The PIC32 MCU stores the reading as a digital value and uses it to control a cursor on an LCD display screen. A program on the PIC32 continually displays obstacles that move across the screen, and the player moves his or her eyes to control the cursor and avoid obstacles.

This system is entirely powered without connection to an AC power source, instead using a 9 V battery to provide power for amplification and a chargeable power source

FIGURE 1

Electrode placement for recording. An Ag-AgCl (silver-silver chloride) electrode was placed at each of the labeled points. Points A and B record the EOG signal for the right and left eyes, and point C provides a ground reference.



**FIGURE 2**

Characterization of EOG signal. An example signal output is shown for a gain of approximately 885.

to power the PIC32. This choice of a power source was important, because it enforces necessary safety considerations for biomedical recording. Connecting a high voltage source to a human user and accidentally completing a circuit path to AC ground could result in serious injury, so great care was taken to use battery power for this project.

A secondary oscilloscope program was also necessarily designed to satisfy a key safety need: The ability to view the recorded EOG signal and test the recording hardware while the circuit is isolated. A normal oscilloscope cannot be used for this purpose for the reasons stated earlier. Care was also taken to apply and fasten the electrodes properly before every session.

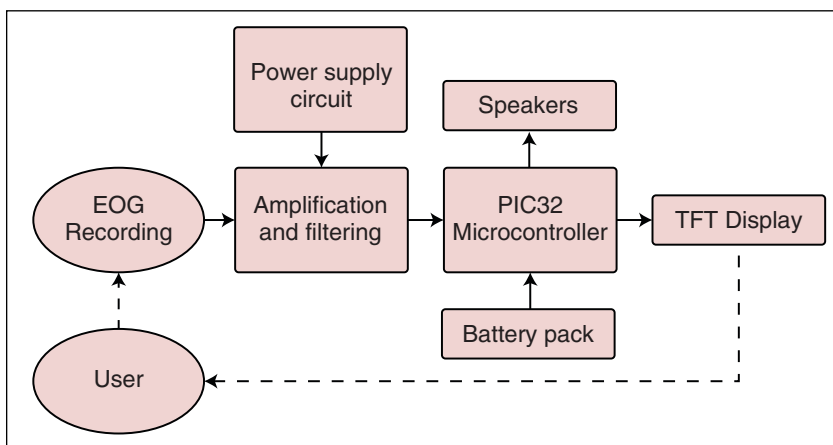
RECORDING AND AMPLIFICATION

Three Ag-AgCl (silver-silver chloride) electrodes are placed around the eyes using a skin-safe adhesive gel—one beside each eye, and one on the forehead as a ground reference—at points A, B, and C respectively, in Figure 1. These electrodes provide the gateway between the biological signal and the digital world, detecting the voltage generated by ions at the skin surface and transducing it into an equivalent electron-based signal.

This voltage is generated directly at the eye, and has some attenuation through the skin surface. A typical magnitude of the raw EOG signal is several millivolts. The voltage readings from the two eye electrodes are

sent to a Texas Instruments (TI) INA121 differential amplifier, which amplifies the difference between the two input signals. This yields a negative or positive voltage based on direction of eye movement. The INA121 provides low noise, a high common-mode rejection ratio, and is suitable for the high-input impedance requirement associated with recording biological signals. **Figure 4** shows the full schematic of the implementation.

A second amplification stage using a TI LM358-based balanced subtractor configuration provides further amplification. This stage reduces the DC voltage component output from the differential amplifier, while

**FIGURE 3**

System overview. "Eye recording" is accomplished with the raw electrode signal.

further amplifying the difference to a range of 0 to 3.3 V—the scale allowed by the PIC32 MCU's on-chip ADC. The resulting signal is a voltage centered at approximately 1.6 V when the user looks straight, with about a 1 V increase or decrease when the user looks left or right, respectively.

This circuit includes several RC components for filtering. It is most important for an EOG recording system to be able to detect the frequency of eye motion—primarily about 0-15 Hz, but with components up to 100 Hz [1]. A system should be able to filter lower-frequency noise present in biological systems (typically 0.5 Hz to 30 Hz) and noise from surrounding electrical systems at about 60 Hz, while preserving information of interest below this frequency range [1]. Therefore, low-pass, 2-pole filter elements with time constants of 0.01 seconds (for a cutoff frequency of 16 Hz) are used to reduce noise outside the intended range for recording, at both inputs to the circuit and at the differential amplifier's feedback component.

A third op amp is used in a unity gain configuration to provide a power bias to our amplifier circuit. The op amp is used to split a 9 V battery into a +4.5 V line and a -4.5 V line for the amplifier. It provides both positive and negative power supply rails for our differential amplifier, and allows for detection of a large voltage swing above or below 0. The unity gain buffer configuration also allows us

to use the op amp's low output impedance to create a low-impedance ground.

BALANCED SUBTRACTOR STAGE

The balanced subtractor stage is also needed in the design to reduce DC offset in amplification. The magnitude of the raw EOG voltage being recorded can differ between individuals, and this difference will manifest as a large, relatively unpredictable DC offset at the output of the differential amplifier. The electrodes used for recording also have an inherent DC offset that can slowly drift during long-term use. The purpose of the balanced subtractor stage is to eliminate this DC offset from our amplified signal, using a voltage reference that is provided as the subtractor's second input using a potentiometer.

Manually tuning this potentiometer to provide the right voltage and negate the DC component allows the device to capture only the difference resulting from eye movement at a magnitude appropriate for the ADC. The ADC voltage range is 0 to 3.3 V. The subtractor allows us to center the EOG signal at 1.6 V and encode left and right eye movement as an increase or decrease from this baseline value. The flexibility of the potentiometer allows the game to be calibrated for different users, or even as the quality of the recorded electrode voltage changes over time for a single user. Using this configuration, the signal and calibration appeared stable for at least several minutes.

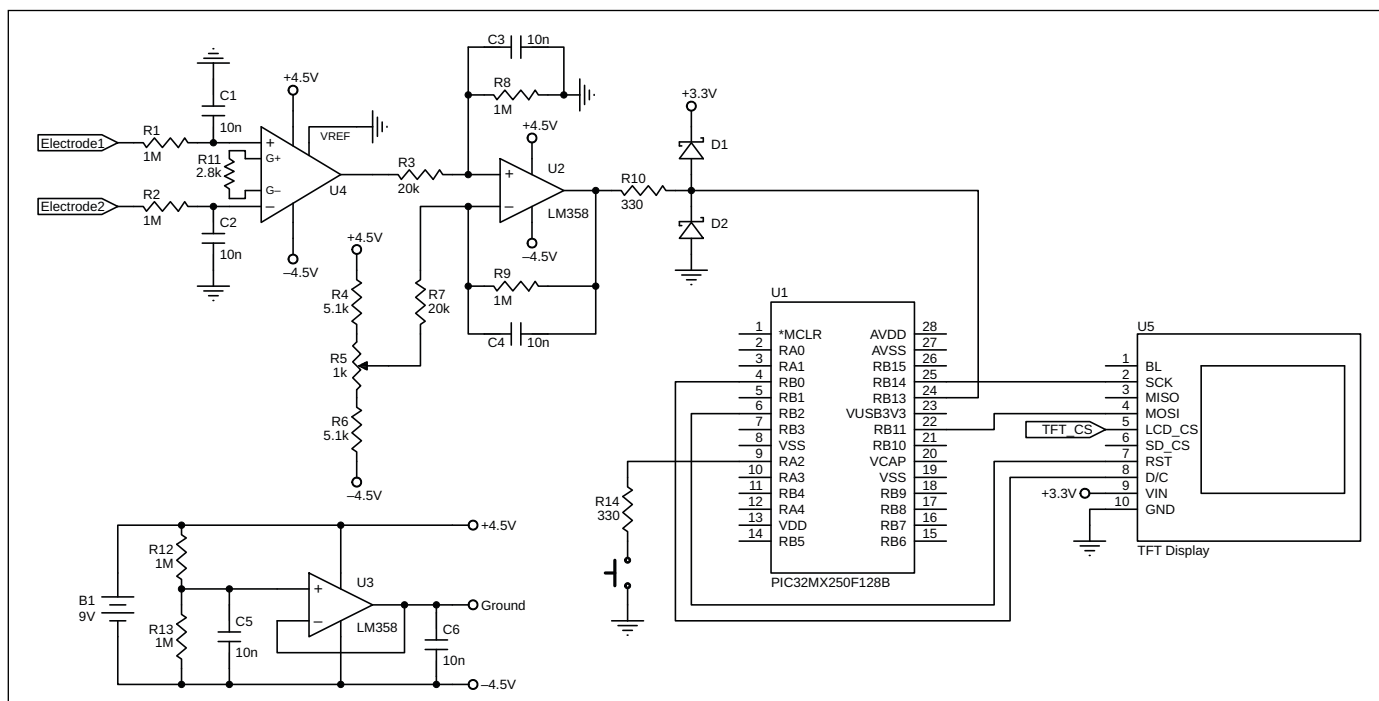


FIGURE 4

Shown here is the full schematic of the system. In the top-left circuit, the Texas Instruments INA121 amplifier represents the differential component. The TI LM358 comprises the subtractor stage. The potentiometer seen at the differential input to the LM358 is used for calibration. The circuit on the bottom-left is used to provide the power source.

The difference mode gain of the first stage in this implementation was 17.7. The gain of the second stage was 50, yielding a total gain of 885. Two Schottky diodes were used to restrict the voltage to between 0 and 3.3 V as a safety measure for the ADC pin on the PIC32.

The EOG recording system described can be customized and tuned to change the range of the amplified voltage or tolerate higher precision. The calibration sensitivity is determined by a voltage divider, shown in the circuit in Figure 4, where the potentiometer is connected in series with two 5.1 k Ω resistors. If the two 5.1 k Ω resistors are replaced with higher-valued resistors, the DC voltage range determined by the potentiometer is smaller, and the offset can be controlled more precisely.

SOFTWARE DESIGN

To use and test our EOG signal, we developed a video game and oscilloscope test program on the PIC32. The video game was shown on a TFT LCD screen. Its components include a bar-shaped cursor at the bottom of the screen that the user controls via eye movement, and randomly generated obstacles that start at the top of the screen and make their way to the player (Figure 5). The player's goal is to use eye movement to avoid the oncoming obstacles, scoring points for each obstacle that passes, until the player errs or the game ends after a predetermined time.

The player control system is divided into three discrete regions by thresholds, so that looking center keeps the cursor in the center of the screen, looking left moves the cursor to the left side of the screen, and looking right moves the cursor to the right side of the screen. The obstacles also spawned in one of these three parts of the screen, moving toward the bottom at a constant velocity.

Before starting the game, a calibration stage also allows the user to tune the potentiometer and make sure the control system works properly. During this stage, the user looks at the center of the screen while slowly adjusting the potentiometer. At the correct DC offset, the cursor will appear at the center of the screen instead of one of the sides, and looking left and right will move the cursor properly as described above. This step typically takes only a few seconds, and should be repeated before every game to account for electrode drift.

The game software consists of three concurrently running threads, using the open source ProtoThreads library [2]. The first is the `protothread_controller{} function`, which periodically updates the position of the user's cursor via the EOG signal provided by the

ADC. This function simply places the cursor in one of three locations on the screen, using two thresholds for the ADC value to determine whether the user is looking right, left or center. Specifically, the EOG signal is centered at about 1.65 V, with a range of 0-3.3 V. This signal is digitally converted to a range of 0 to 1,024, centered at approximately 512.

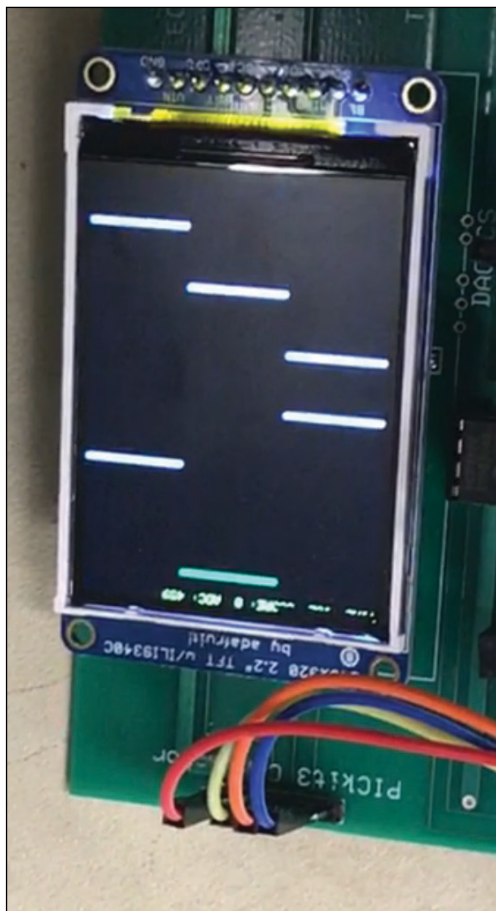


FIGURE 5

Photo of the running game. The user controls the bar at the bottom, and the other bars are the obstacles that approach from the top of the screen.

ABOUT THE AUTHORS

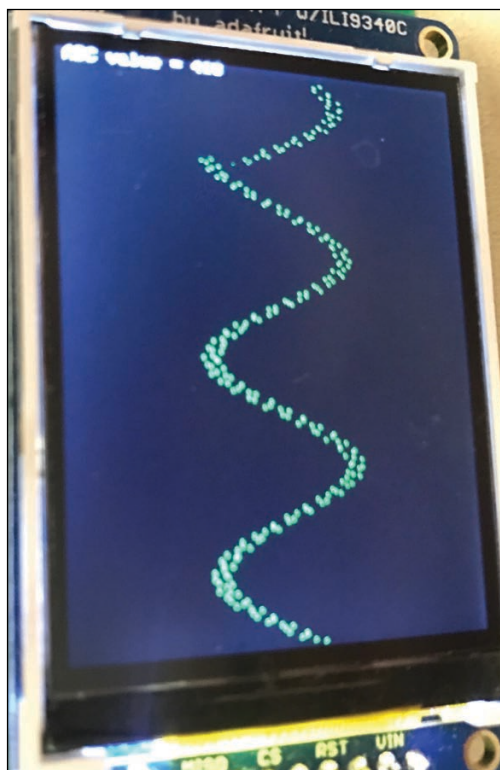
Eric Cole is now a Bioengineering PhD student at Georgia Tech, having graduated from Cornell University in May 2018. His research in the Department of Neurosurgery at Emory University applies electrical engineering principles to evaluate and develop novel stimulation methods to treat epilepsy. He also enjoys playing piano and composing musical theatre and jazz.

Alex Huang graduated from Cornell University in May 2018 with a B.S. in Electrical and Computer Engineering, and is currently a Software Engineer at DHPC Technologies. His technical interests include programming in C/C++. During his free time, Alex enjoys going to the gym and DJing.

Evan Mok graduated from Cornell University in May 2018 with a B.S. in Electrical and Computer Engineering, and is currently a physical design engineer at IBM. His technical interests include VLSI design and computer architecture. During his free time, Evan enjoys playing volleyball, bouldering, and road tripping.

FIGURE 6

Oscilloscope program image. This signal shows an amplified sine wave test signal, generated via a function generator and recorded by our oscilloscope program. Test signals consistently showed fidelity and minimal impact from noise.



We set two thresholds at 700 and 300, so that significant eye movement away from center moves the cursor to the left or right side of the screen. This threshold scheme appeared to perform consistently, and avoided any problems arising from finer noise or oscillations in the EOG recording. But these threshold values could still be adjusted as needed, to be closer to or farther from center for greater or lower sensitivity. Once calibrated properly, the game is begun by pressing a button—activating a Boolean variable in software using a pull-down resistor to launch the game.

The second thread, `protothread_dynamics{}`, randomly spawns obstacles every few frames and updates locations as they progress toward the user. It also determines whether a collision has occurred, and ends the game if the condition is met. The obstacles are stored as an array of structs with X and Y coordinates, using pixels on the screen as Y

values and 0, 1 or 2 as X values to indicate the horizontal position. A valid bit indicates that the obstacle should be displayed on the screen.

Every 67 ms, this thread generates a new obstacle, using a pseudorandom number generator to choose the X position. It updates the Y position so that all active obstacles move towards the user's cursor in Y. A seed for the pseudorandom number generator is set based on the time of the game-starting button press, essentially making the random sequence different on every play. On each iteration, the thread also checks if the user's X and Y positions are the same as one of the obstacles, setting a "game over" variable if so, enabling the timer thread to display a "game over" message.

The third thread, `protothread_timer{}`, is a simple timer thread that updates a 1-second tick counter on the game screen, display the game score, and end the game (printing a "Game Over" message) once time is up or a game-ending collision has been detected.

A second program was developed to provide an oscilloscope function. It plots the waveform of the recorded signal over several seconds, which is useful for confirming that the recording is working properly. Information on the software resources we used, including details and resources on borrowed graphics libraries is available on the *Circuit Cellar* article materials webpage [3]

RESULTS

The final running system (**Figure 5**) successfully used EOG to control a video game in which the control was heavily separated into thresholds, rather than in a completely continuous control scheme. While use of a continuous control scheme was not specifically tested, the sequence described above appeared to display very stable behavior, and would likely have operated properly if the screen had been divided into smaller subsections.

At a small enough scale, it is difficult to determine whether noise interferes with the PIC32's ability to determine fine changes in the EOG signal, or whether volatility in the control scheme is due to the user's own inability to control eye movement precisely. The eyes constantly flicker and produce sporadic movement, introducing noise in and of themselves. However, the EOG clearly has potential for a greater level of precision than what was demonstrated in this project, as indicated by a typically clean oscilloscope signal, even when testing on a real player.

This project implemented only horizontal EOG recording, but electrodes can also be placed above and below the eye to record two-dimensional EOG. Such a control

For detailed article references and additional resources go to:

www.circuitcellar.com/article-materials

References [1] through [3] as marked in the article can be found there

RESOURCES

Adafruit | www.adafruit.com

Microchip Technology | www.microchip.com

Texas Instruments | www.ti.com


system could open many more possibilities for inclusion in different software systems, with only the added cost of implementing a second amplification circuit with the same configuration as the first. **Figure 6** shows a signal shows an amplified sine wave test signal, generated via a function generator and recorded by our oscilloscope program.

The EOG recording method provides an inexpensive, accessible, and portable way of using biomedical signals to control an electronic device or simply record eye movement. One aspect of implementing this scheme is to calibrate it to accommodate different users. The calibration method for this implementation was effective, but still manual. We examined the user's control of the cursor before the game began, and manually adjusted the potentiometer until it was as expected (the cursor moved left, right and center when the user looked in each respective direction. This adjustment corresponds to proper setting of the subtractor DC input, biasing the recorded signal to center at the middle of the ADC's voltage range.

Over the course of a given session, the EOG signal displayed drifting, and it did not maintain a constant magnitude over time. This was necessarily due to the instability of the electrodes used, which dry out and

lose the integrity of their connection. This can occur when the gel dries over time, shorting the path between the electrode and skin. Therefore, if accuracy is important, this method of implementing EOG may not be suitable for long-term usage without replacing and reaffixing electrodes.

The calibration process could be automated or improved in various ways to adjust for these complications. For example, a combination of a variable resistor controlled by software and sufficient programming to determine this value could result in a completely automated system that accommodates different users. If the EOG signal also doesn't rail to a single voltage value, a program can be used to set threshold values automatically that correspond to differences in signals between individuals.

This affordable and effective way of recording eye movement could be safely introduced in many applications. EOG could improve the development of virtual reality devices, greatly increasing ease of use and cost in comparison to other methods—such as use of cameras and computer vision techniques to interpret eye movement. EOG could also find effective use in supplementing motor control with prosthetics for any devices that heavily utilize a single degree of freedom in motion—wheelchair rotation, for example. 

LOOKING TO ADVANCE YOUR CAREER?

Be a part of one of the
**top Electrical Engineering
programs in country**
and experience the
Bearcat Promise!


University of
CINCINNATI | ONLINE

online.uc.edu

Fall registration is
open now

Macros for AVR Assembler Programming

Tools of the Machine Code Trade

FEATURES

The AVR microcontroller instruction set provides a simplicity that makes it good for learning the root principles of machine language programming. There's also a rich set of macros available for the Microchip AVR that ease assembler-level programming. In this article, Wolfgang steps you through these principles, with the goal of helping programmers "think low-level, write high-level" when they approach embedded systems software development.

By *Wolfgang Matthes*

In today's modern world, why program in assembler? If you program for a living, there are many factors that drive which programming language you use. You could use a language that's readily available, or one that your customers desire or a language that your superiors have mandated. Or, you could use a language that's currently in vogue within a certain communality such as education. But, whatever your programming language of choice is, there will be always compelling reasons to program a machine down to the metal. Those who can't program at the machine level do not really know how a computer works. Moreover, having a familiarity of the basic machine code level of programming is a prerequisite to being able to write truly efficient high-level code. Yes, compilers generate machine code. But you should be able to read it and to judge its efficiency.

By having a notion of what the machine code produced by your high-level code statements look like, you'll be able to write programs that run faster and require less memory space. In short, it's wise to follow the advice right in the title of a book by Randall Hall called: "Thinking Low-Level, Writing High-Level" [1]. When writing programs for embedded systems—device drivers and the like—occasionally it can be an actual necessity to resort to assembler programming. That's typically the case if tight timing requirements are to be met. Sometimes, you have to deal with microseconds or even machine cycles. In such cases, it's not advantageous to rely on a compiler. Finally, it goes without saying that you could dive into assembler programming projects just for fun.

THE ROLE OF MACROS

Experienced craftspeople select and prepare their tools before they start working. Sometimes, they make their own special devices to do their work—things like gauges, stencils or rigs. Macros may be likened to these kinds of tools for us programmers. Essentially, macros constitute a toolbox. Some macros simply ease program writing. Others are provided to circumvent some shortcomings of the architecture. Let's suppose that our projects aren't that big, our memory requirements are not that severe and absolute speed isn't a top priority. Under these conditions, we can afford to execute multiple instructions instead of one instruction of a more advanced architecture.

The basic idea is to create a simple virtual machine and to define an extended instruction set, which alleviates typical programming chores—like querying and modifying single bits, 16-bit operations, conditional branching and stack-based operations. In this article, we will introduce the virtual machine and illustrate the content of our toolbox. Here we will concentrate on the basics. In the beginning, it is important to get a first impression, an overview of principal tools and methods. A comprehensive description, the source code and application examples can be found on the Internet.

Many instructions and nasty details: Those words summarize the principal impediments of programming a processor down to the metal. In itself, assembler programming isn't that difficult. However, when the programming task is more demanding, an overwhelming number of

instructions have to be written. That's especially true for RISC and minimalist CISC architectures, which require many instructions even for small operations.

For example, instead of simply moving a memory operand to a peripheral unit, a RISC machine requires you to load it first into a register. At the same time, we have to master the complexity of the machine architecture's principles of operation, including their restrictions and quirks. To mention a few examples, not all instructions will work with all registers and all types of operands. And some addressing modes have very short address offsets that may, for example, only be able to jump over 64 instructions or to access only 64 bytes.

How do you overcome those restrictions? The most obvious solution would be to program solely in higher-level languages, leaving it to the compiler author to cope with such eccentricities. However, we don't want to tread that path. The other extreme would be to design a new processor. Thanks to FPGA technologies and hardware description languages, that wouldn't be completely impossible with today's technology. Nowadays, it could be done even on the proverbial kitchen table.

A typical run-of-the-mill processor core is not that difficult. An 8-bit RISC CPU core would be merely some kind of a student's assignment (refer to [2] and [3] as examples of appropriate textbooks). All that said, when your homemade processor core is up and running, it will lack advanced peripherals, not to mention the absences of any rich ecosystem. Therefore, that option is only a viable approach if done for research, educational purposes or just for fun.

VIALE ALTERNATIVES

A feasible solution must be based on a well-proven architectural platform and a readily available integrated development environment (IDE). Our goal is to ease assembler programming. This is achieved by substituting cumbersome instructions or tedious instruction sequences with virtual instructions that are more powerful and easier to use. The application programmer writes down one instruction and the machine executes a short program showing the same behavior. Essentially, there are three principles involved in laying out such substitute programs: subroutines, macros and emulation.

Functions: Programs written once and called whenever needed are known as functions, procedures, methods and the like. Consider this simple generic example. Our application example relates to a display unit—think of an LCD or OLED module or even of a window on a PC screen. We want to display a character string at a certain screen position, given by the X and Y coordinates. What could be more natural than providing somewhat similar to the following?

```
SHOW_TEXT (X, Y, string_pointer) or SHOW_LITERAL (X, Y, plain_text)
```

The program itself has to be declared and written. This is the so-called procedure body. (For our examples, we orient ourselves by the Ada programming language. Although the syntax is more verbose, it is also more lucid and instructive than C, Java and the like.)

```
procedure SHOW_LITERAL (X: in INTEGER; Y: in INTEGER; plain_text: in ASCII_string) is
begin
```

Here is the program that does the work ...

```
end SHOW_LITERAL;
```

When such substitute programs have been made available to an application program, it is easy to invoke them:

```
SHOW_TEXT (message_X, message_Y, I/O_error_message);
```

```
SHOW_LITERAL (2, 20, "Temp out of range");
```

A substitute program must know which data to work with. Those data will be passed by parameters. Whoever writes the program body must declare the parameters. In our example procedure `SHOW_LITERAL`, this is done in the first lines. And whoever uses such a program inserts the current parameters. They must be of the right type, must be within allowed ranges and so on. When programming in a high-level language, the compiler will check whether the parameters are correct or not. When programming in assembler, it is solely up to the programmer to avoid errors.

Subroutines: The subroutine is the counterpart to the function, procedure or method in a high-level language. A subroutine is stored only once. To be used, it must be called. In most cases, a high-level language function, procedure or method will be translated to a subroutine. In this process, the compiler will do the housekeeping work. However, when programming in assembler, parameter passing, calling

and returning to the calling program have to be coded instruction by instruction. It goes without saying that there are no formal parameter declarations and correctness checks. Returning to our examples, a subroutine `SHOW_TEXT` begins with a label and ends with a return instruction:

```
SHOW_TEXT:
```

Here is the program that does the work ...

```
RET      ; Return to the calling program.
```

To be invoked, the parameters are passed:

```
LDI r16, x
LDI r17, y
LDI z1, low(text_adrs)
LDI zh, high(text_adrs)
CALL show_text
```

The prerequisites of this example are that the assembly language for the Microchip Technology (formerly Atmel) AVR microcontroller (MCU) is used and that the parameters are passed in the AVR register file. There are different principles of parameter passing. They will be explained thoroughly in the textbooks of assembler programming. [4] to [6] are typical examples.

Macros: A macro is a program sequence that is written once but inserted always when invoked. The parameters are inserted by the assembler. Parameter passing is a very basic mechanism, without formal declarations and even—as in the case of the AVR assembler—without particular mnemonics. In the macro body, the parameters are simply numbered by @0, @1, @2 and so on. A macro body of our example could look like:

```
.macro SHOW_TEXT
LDI r16,@0      ; @0 = coordinate X
LDI r17,@1      ; @1 = coordinate Y
LDI z1, low(@2) ; @2 = string address
LDI zh, high(@2)
```

Here is the program that does the work ...

```
.endmacro
```

Invocation is straightforward because the parameters have been already inserted by the assembler:

```
SHOW_TEXT example_x, example_y, example_string
```

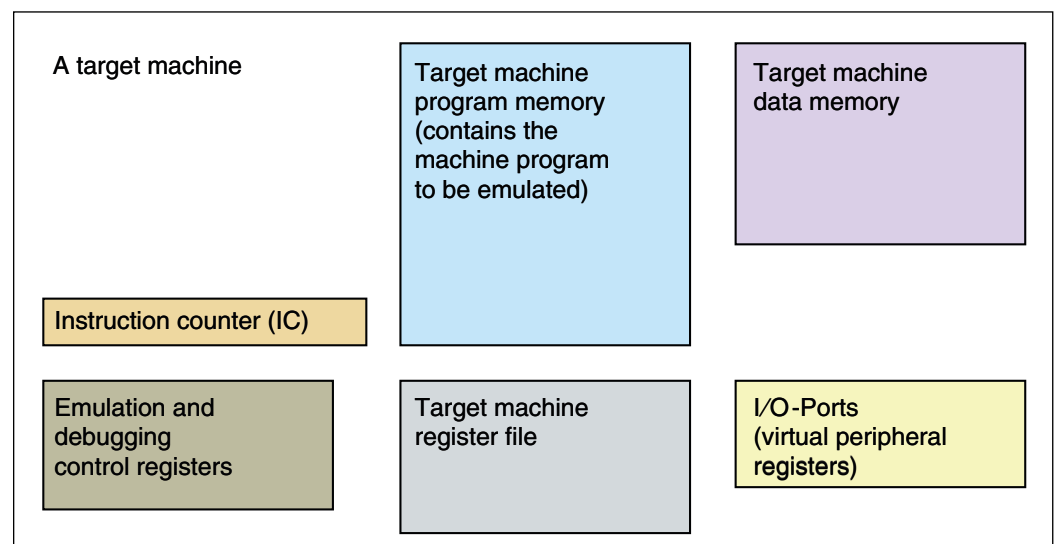


FIGURE 1

Data structures of a typical emulator. Each target machine requires a set of such data structures.

A macro will occupy much more memory capacity (the whole substitute program instead of some instructions to pass the parameters and to call the subroutine). However, it will run faster because there is no overhead to pass the parameters, to call the subroutine, and to return to the calling program.

If the substitute program is comparatively large—showing text on a display is a good example—it may be advantageous to write the program that does the work, as a subroutine and to provide macros to invoke it conveniently:

```
.macro SHOW_TEXT
    LDI r16,@0          : @0 = coordinate X
    LDI r17,@1          : @1 = coordinate Y
    LDI z1, low(@2)      : @2 = string address
    LDI zh, high(@2)
    CALL EXEC_SHOW_TEXT ; Call the program that does the work
.endmacro
```

The program will be invoked by the statement: `SHOW_TEXT x, y, string` as shown above.

Inserting the machine code of a substitute program instead of calling it avoids the overhead of parameter passing, subroutine call and return. Many compilers support this option. For example, a statement `pragma inline` will cause the compiler to implement and invoke the function or procedure as a macro.

Emulation: An emulator is a program that implements the functions of a processor architecture by software. The machine running the emulator is the host. Its architecture is the host architecture. And the architecture to be emulated is the target architecture. The target instructions are not invoked by instruction fetches of the host machine. Instead, these instructions are treated as data structures, which the emulator program addresses. The other architectural features of the target architecture are represented as well with stored data structures, especially arrays (**Figure 1**).

In principle, an emulator is a fairly simple program loop (**Figure 2**). The program to be

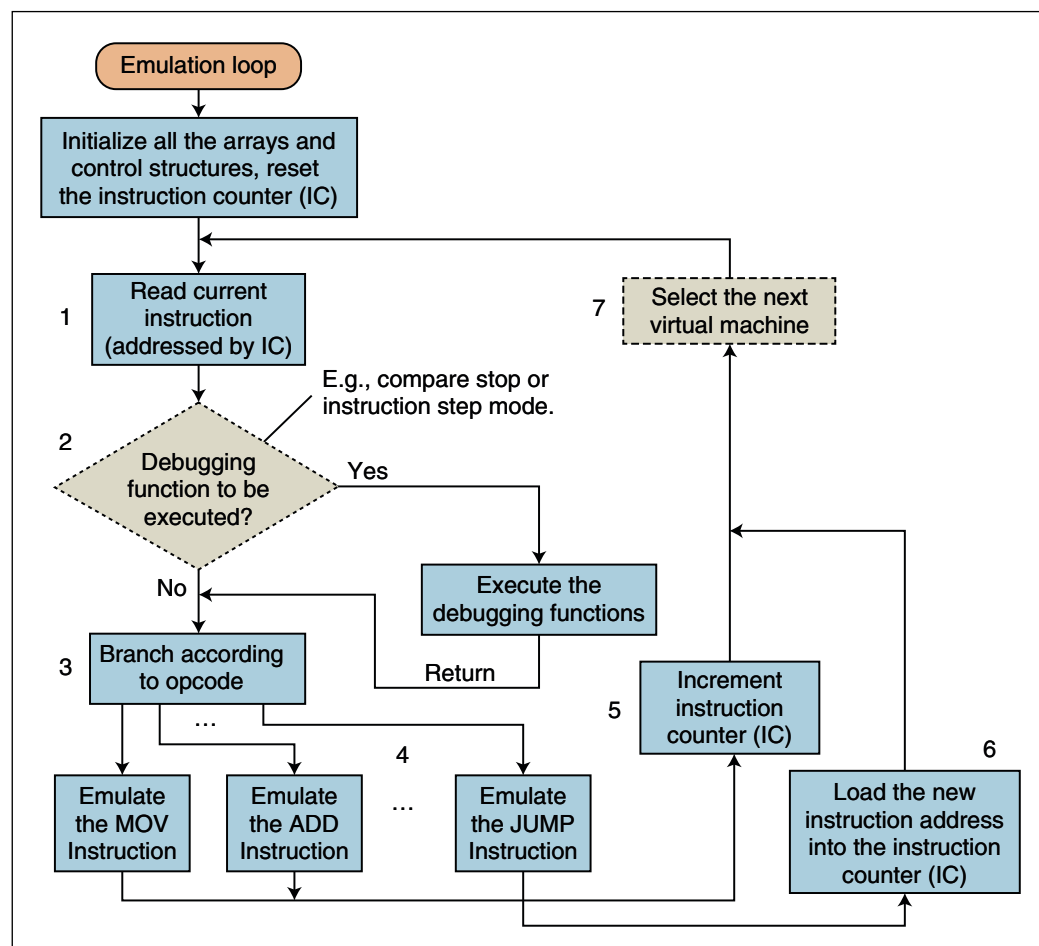


FIGURE 2

A typical emulator loop is shown here as a flowchart. (1) Fetching the instruction of the target machine. The program memory array (see Figure 1) will be addressed by the target machine's instruction counter (essentially, a stored integer variable). (2) Debugging functions are executed, if so desired. (3) To invoke the emulation routine of the particular instruction, the instruction's operation code is used to address a branch table. (4) Each of the target machine's instructions has its own emulation routine. (5) If the current instruction is no branch or call, then the instruction counter will be incremented to address the next instruction. (6) Branch, jump and call instructions will load a new instruction address into the instruction counter. (7) If the emulator supports multiple virtual machines, the next virtual machine to serve will be selected.

emulated is loaded into the program memory array. The emulator fetches instruction for instruction out of this array and invokes routines reproducing the effects of the particular instructions. Emulation allows you to implement even unconventional, eccentric target architectures. It's an affordable method to tinker with your own architectural ideas and instruction sets. A particular strength lies in the realm of debugging, error tracing and the like. Because everything runs by software, all the details of the program behavior can be analyzed. Even the most severe errors in the target program will not crash a well-written emulator.

Furthermore, emulation can easily support multiple virtual target machines. It's only necessary to provide the data structures shown in Figure 1 and to switch the emulator loop appropriately. This is already depicted in Figure 2 (block 7). All that said, there is a principal downside: Emulation is slow. A typical target instruction will require from about 10 to more than 50 host instructions. So, it only makes sense to resort to this principle when such lower speeds aren't an impediment.

Why concentrate on macros? It's because our focus is to get application projects up and running. With this goal in mind, there is no reason to take time to bother with instruction sets, writing our own assembler and the like. We must rely on a well-established ecosystem. That's why we're focused here on the assembler language of a chosen MCU family—the AVR—and supplementing it only by some more advanced, convenient or specialized tools. Moreover, we need the speed of our MCU. For all those reasons, emulation is out of the question.

ABOUT THE AUTHOR

Wolfgang Matthes has developed peripheral subsystems for mainframe computers and conducted research related to special-purpose and universal computer architectures for more than 20 years. He has also taught Microcontroller Design, Computer Architecture and Electronics (both digital and analog) at the University of Applied Sciences in Dortmund, Germany, since 1992. Wolfgang's research interests include advanced computer architecture and embedded systems design. He has filed over 50 patent applications and written seven books. (www.realcomputerprojects.dev and www.controllersandpcs.de/projects).

SELECTING CANDIDATES

It goes without saying that experienced application programmers will write subroutines and macros whenever they see fit. Our primary goal, however, is to provide something useful in advance. One of the more specific goals could be a specialized application programming interface (API), for example, to cope with Boolean expressions or to support LCD display modules. Therefore, we have to ask, what the most basic functions are: like positioning to an X-Y-coordinate, showing a character string, or drawing a line.

Another class could be API functions to support application programming in general. A typical example is a set of CISC-like or stack-oriented virtual instructions to ease assembler programming of RISC machines. Macros or subroutines can also capture the experience collected from project work. For instance, when a project is reasonably large and programming experience grows, we will recognize instruction sequences that appear again and again.

There are also functions that are noticeably difficult to program, so we want to encode them only once. Short sequences are obvious candidates for macros. When a program is somewhat larger and more complicated, it is often advantageous to implement it as a subroutine and to provide supplementary macros to call it. That relieves the application programmer from the tedious programming chore of parameter passing.

The small virtual machine is a principle that's useful when assembler-programming each category of architectures. It will relieve the programmers of the restrictions of an outright minimal register model MCU architecture as well as of the complexity of a high-performance processor. While in the majority of applications an above-100-MHz Arm or MIPS will be programmed exclusively in high-level languages, there are some occasions when one would prefer not to rely on a compiler, but rather to program down to the metal.

Typical examples are time-critical device drivers or innermost loops of really performance-hungry applications. With respect to ease of assembler programming, RISC architectures are somewhat infamous. How do you keep track of variables in 16 or 32 registers? So, it could be wise to define a comprehensible small virtual machine with only a few registers and to create a CISC-like programming environment by writing appropriate macros. Besides ease of comprehending, there is another reason to keep the virtual machine small and simple: It should not use up all the processor's resources (like the complete register file). This will allow

For detailed article references and additional resources go to:

www.circuitcellar.com/article-materials

References [1] through [7] as marked in the article can be found there.

RESOURCES

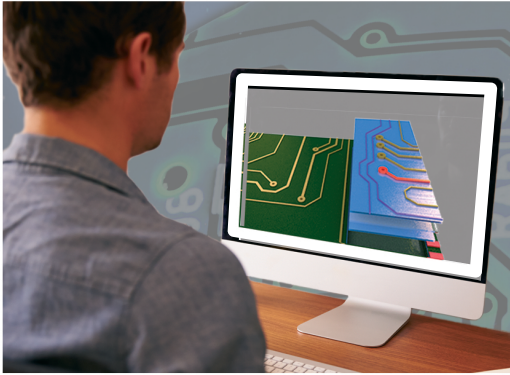
Microchip Technology | www.microchip.com

PROTEUS DESIGN SUITE

Advanced PCB features for professional board design. Try it today!

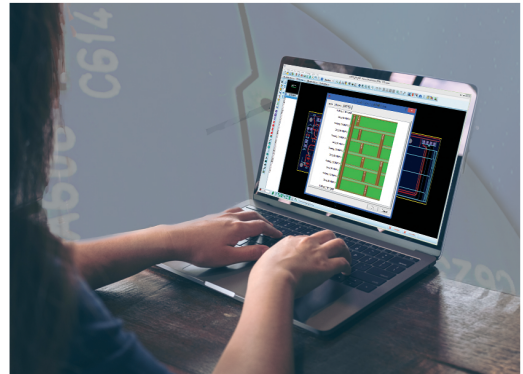


DESIGN ROOMS



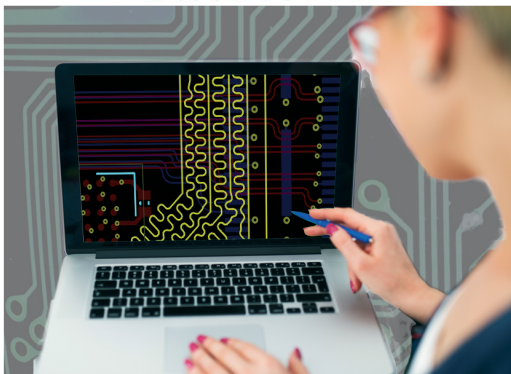
Set design rules that apply in user specified areas of the PCB.

LAYER STACKUP



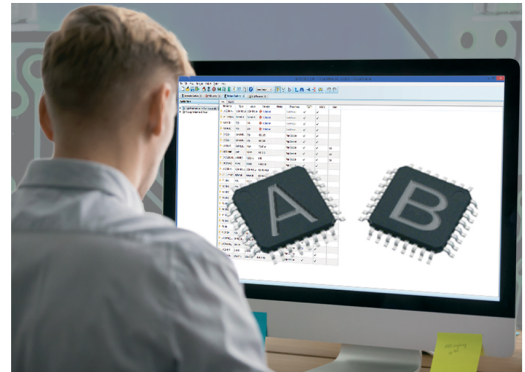
Control the layer stackup and drill ranges for smarter routing.

SERPENTINE ROUTING



Easily length match tracks against each other or to a target distance.

DESIGN VARIANTS



Edit the fitted status of parts or replace with pin compatible alternatives.

**Now includes support for EDIF2:
Import your OrCAD Schematics directly into Proteus.**

Visit: www.labcenter.com

Toll Free: 866.499.8184

Tel: 905.898.0665

E-Mail: don.jackson@labcenter.com

youtube.com/c/LabcenterElectronicsLtd

labcenter  www.labcenter.com
Electronics

resorting to conventional programming, whenever necessary.

It is even possible to collaborate with programs compiled from high-level languages. Compilers impose typical restrictions, for example, which registers the application programmer should not touch. They should be observed carefully. Nevertheless, our own virtual machine has been designed with disregard for this wisdom, because it has been thought of as a purely experimental project.

WHY AVR ARCHITECTURE?

AVR MCUs are ubiquitous and inexpensive. The instruction set is comparatively well-suited to learn the basics of machine programming. After a few hours, you can have positive first experiences. In other words, it is highly probable that your own first programs will run. AVR is neither as complicated nor as difficult

to program as a 32-bit architecture, nor as minimalist as alternative 8-bit architectures. In contrast, the AVR architecture is somewhat similar to the more advanced RISC architectures. That the AVR is an 8-bit architecture, is not that important.

Any arbitrarily small universal processor can execute any algorithm, provided the storage capacity is sufficient and execution time does not matter. “Turing-completeness” is the technical term describing this fact. For most educational and fun projects, problem complexity and size are rather modest. We talk about projects that a single person can tackle within a few hours to a few weeks. For this purpose, 8-bit processing with a clock speed between 4 to 32 MHz should be good enough—especially given that an AVR requires only a single clock cycle for most instructions. Keep in mind, that the computers aboard the

Rank	Architectural peculiarity
1	The flag bits are accessible only via the I/O address space. It is awkward to save and restore the flags. Both operations are required frequently. Hence, other architectures provide dedicated instructions for this purpose (for example, an instruction pair PUSHF and POPF).
2	The stack pointer (SP) is accessible only via the I/O address space, impeding stack manipulation and stack-relative addressing with the SP as a base address register.
3	Only half of the register file can be operated upon by immediate operands. So only 16 registers are true general-purpose registers.
4	There are only three address registers (X, Y, Z). To boot, they belong to the general-purpose-registers mentioned above. In consequence, there are only 10 register bytes freely available, which can be operated upon by immediate operands and in which individual bits can be set and queried. Three address pointers, base registers, index registers or the like are not enough. When speaking of a true general-purpose register file, it would be best when each register could be used for addressing. At least, 6 to 8 address registers would be a sufficient minimum.
5	The instructions to set and clear bits in the register file (SBR, CBR) are only aliases of the instructions ANDI and ORI. Their operands are bytes, serving as bit masks. The skip-type instructions to query individual bits (SBRC, SBRS), however, have bit indices. This gives rise to annoying mistakes. SBR r16,0 does not change anything (corresponds to ORI r16,0), whereas SBRS r16,0 queries bit 0 ... Individual bits can be queried in all 32 registers, but set or cleared in only 16 of the 32 registers.
6	There is no uniform I/O address space. I/O instructions can address only 64 registers. Instructions to query and set individual bits can be applied only to the first 32 I/O addresses. If the microcontroller’s periphery requires more than 64 addresses, the additional registers can be addressed only by memory access instructions.
7	The branching conditions related to arithmetic comparisons by subtracting ($A - B$) do not correspond completely to the usual industry standard. For some programming intentions, one has to subtract the other way around ($B - A$).
8	The jump distance of the conditional branches (BRANCH instructions) is often too low. In some cases, the traditional combination SKIP and JUMP would do better. However, SKIP can only query the bits of the register file and the first 32 I/O addresses, but not the flag bits in the status register.
9	To move bits around, a dedicated flag has been provided, the T-flag. In contrast, it has been an industry standard for decades to use the Carry Flag (CF) for this purpose. This convention dovetails with the rotate instructions, thus allowing to assemble bytes from particular bits (gather operation) or to dissect bytes and emit the particular bits (scatter operation).
10	The important XOR operation (to selectively toggle bits) is not available with an immediate operand.

TABLE 1

Some peculiarities of the AVR architecture ordered according to their importance, the most severe first.

Apollo spacecraft had less speed and memory capacity than even one of the more advanced 8-bit MCUs of today.

The AVR architecture is noticeably more advanced than other 8-bit architectures. Nevertheless, it has some grievous restrictions of its own. But which of those restrictions are severe and which are merely nuisances? Some conspicuous restrictions are listed in **Table 1**, the most severe listed first. The degree of severity has been judged according to programming experience and in comparison to other well-renowned architectures.

Our virtual machine has three working registers A, B, C and three address registers X, Y, Z (**Figure 3**). These registers consist of two bytes each, which can be accessed separately. Register A is the accumulator. Register B typically receives the second operand. Register C is mainly used for counting and auxiliary functions. The address registers X, Y, Z belong to the basic AVR architecture. Additionally, the macros may access a general working register TEMP, the registers R0 and R1, the stack pointer (SP) and the status register (SREG). The remaining registers of the AVR register file are freely available. The comprehensive documentation, individually describing each macro, is quite voluminous. With that in mind, we will limit the description to an overview (**Table 2**) and supplement it with some details and examples. The documentation and the source code are available for download. Links to them can be found on *Circuit Cellar's* article materials webpage.

An assembler is basically a program that translates character strings—like mnemonics and labels—into bit patterns, like machine instructions, addresses and other memory content. There are so-called meta-assemblers able to create machine-specific assemblers for arbitrarily complicated architectures. Besides cost, the downside is that you have to set up all the translation tables for yourself. Therefore, we will be content with the native AVR assembler.

Our starting point is the assembler 2 (AVRASM2) of the AVR development chain. It's not overly complicated and it's easy to understand. The assembler is built around certain keywords, denoting instructions, registers and built-in functions. Additional mnemonics are defined in device definition (.INC) files, which belong to the particular microcontroller devices. Built-in keywords and pre-defined mnemonics cannot be used as user-defined symbols (in other words, as labels or names in .def, .equ or .set directives).

The .macro directive denotes the start of a macro. The .endmacro directive terminates the macro body. The parameters are denoted by @0, @1 and so on. The AVR assembler 2

does not limit the number of parameters. When invoking a macro, the parameters are passed as mnemonics, labels, numeric values or character strings. A macro definition may invoke other macros. How deep macros may be nested, is not specified. However, Microchip advises not to overuse this feature. According to practical experience, a nesting depth of three should work.

MACRO SYNTAX

The application programmer sees a macro as a new instruction mnemonic. Some macros have no parameters at all. Some have one parameter, some two or more parameters. We have to consider quite a few conventions and restrictions: the built-in keywords, the predefined mnemonics and the capabilities and peculiarities of the assembler's preprocessor. Above all, the assembler does not support the overloading of built-in keywords and predefined mnemonics. As a result, we must circumvent all those mnemonics and create our own. So, for example, we will call a literal—what in AVR terms is dubbed an immediate.

Each word or abbreviation we have made to a mnemonic for our macros, however, cannot be used freely as a label or another mnemonic in the application program. So we should not be too generous. A particular problem is, how to encode different variants of a certain operation. Let us contemplate, for example, how we could denote a macro to load a 16-bit register (A, B, C, X, Y, Z).

On one hand, we could write macros like LDA label, LDB label and so on, yielding 6 different macros. On the other hand, we could

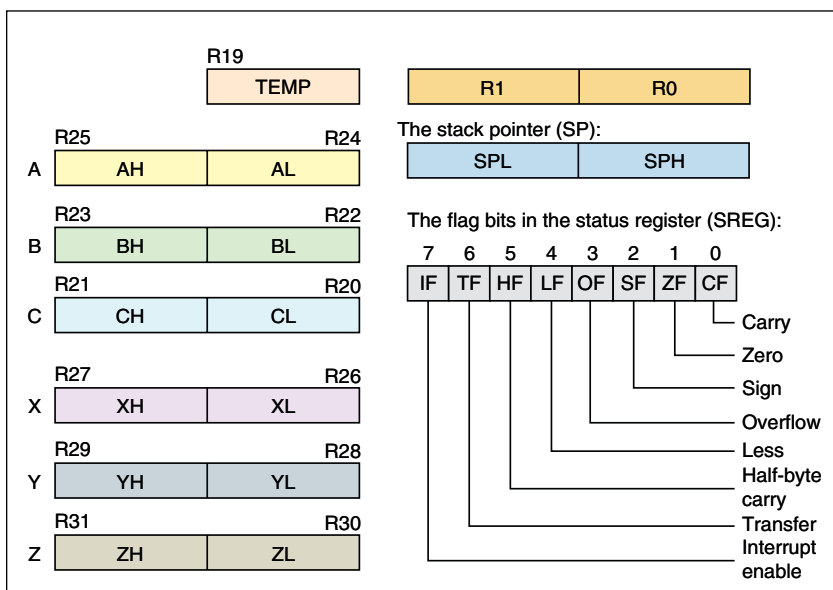


FIGURE 3

Registers and flag bits of the virtual machine. A table supplementing this diagram is provided on Circuit Cellar's article materials webpage.

define one single macro and pass the register as a parameter: `LDW A, label; LDW B, label` and so on.

In the second variant, the macro body would be considerably more complicated, and we have to define the mnemonics A, B and so on by `.equ` statements. Problem is this approach will fail if it comes to X, Y and Z. Those letters are reserved keywords, built into the assembler. Although we can pass a register name to a macro, the preprocessor will not accept register names in conditional statements. So, we had to cram all the different variants into the macro mnemonic, making it occasionally somewhat unwieldy. Nevertheless, this habit is not without exceptions—there are alternatives. For more details, refer to the documentation.

MEMORY AND I/O ADDRESSING

Figure 4 depicts the memory and I/O addressing of different AVR series. All I/O registers are accessible via data memory (SRAM) addresses. A maximum of 64 I/O registers can be addressed in I/O instructions. Single-bit accesses are supported for the first 32 I/O addresses. In the Xmega, the I/O addresses are equal to the data memory addresses. In the ATmega or ATtiny the I/O addresses are higher by 32 (20H). This must be declared (by `.equ` statements) or programmed in the application program.

If the address value is less than 40H or (for single-bit access) less than 20H, it is considered an I/O address and access is performed with I/O instructions. Otherwise, it is a data memory (SRAM) address. AVR I/O instructions support only direct addressing. The address parameter is an immediate. If the programmer wants to address an I/O device with an address parameter held in a register (indirect addressing), the device must be addressed with its data memory (SRAM) addresses. Example:

```
OUT udr, r16 ; Output to the UDR register (ATmega16) via its I/O address
```

```
LDS udr + 0x20, 0x55 ; Output of an immediate value to the register UDR
                     ; via its data memory (SRAM) address
```

When 16-bit registers in I/O units are accessed, the byte order matters. In ATmega or ATtiny versions of the AVR MCU, write into the high-order byte first, read from the low-order byte first.

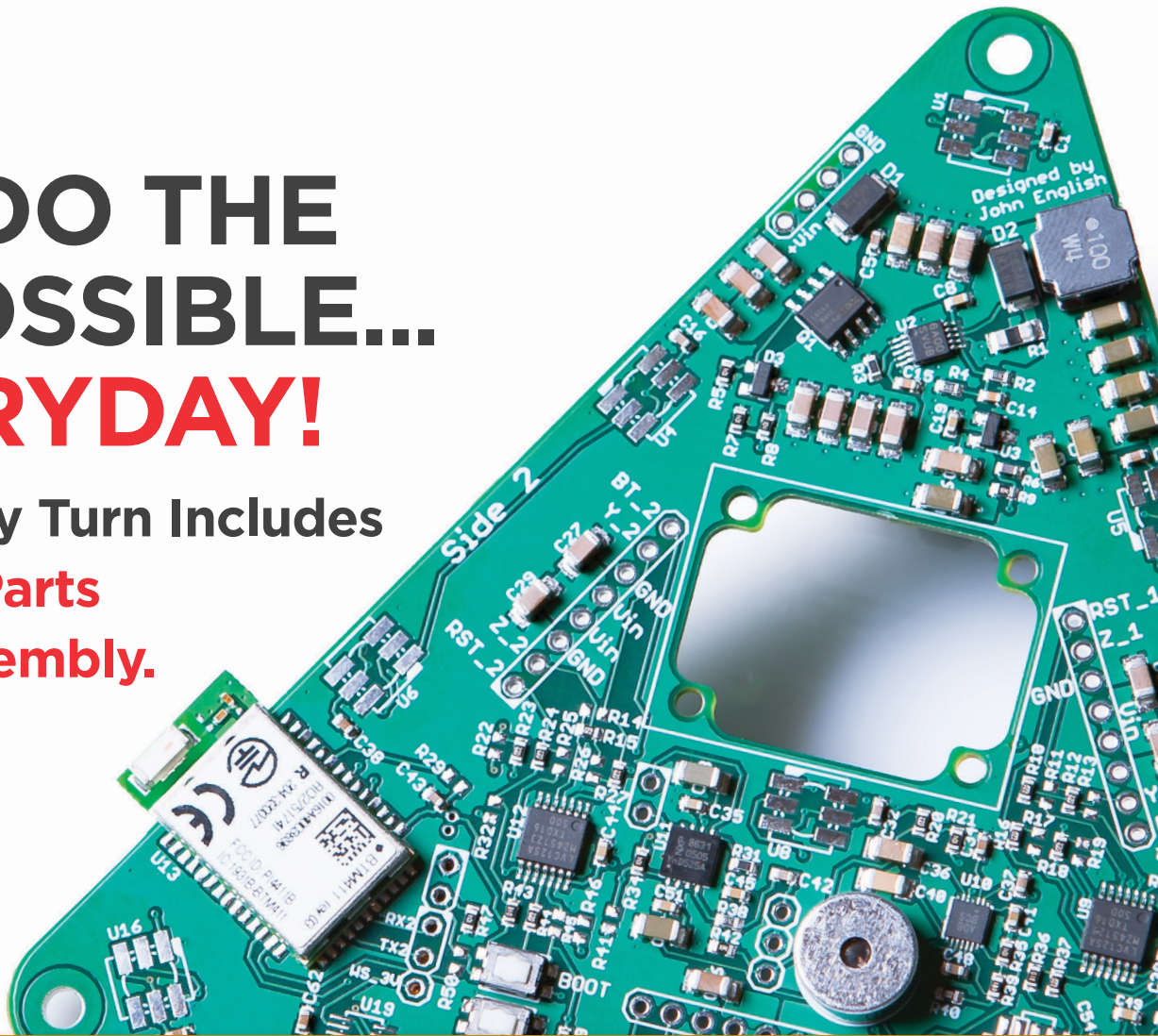
No.	Macro functions	No.	Macro functions
1	Basic transports	15	Load or store a 16-bit word (registers A, B, C, X, Y, Z)
2	Single-bit operations	16	Operations with 16-bit literals (registers A, B, C, X, Y, Z)
3	Jump on a flag	17	Operations with 8-bit literals and registers r0...r31
4	Jump on a single bit	18	Operations between 16-bit registers
5	Jump after subtracting A – B (compare arithmetically)	19	Operations between 16-bit and 8-bit registers
6	Skip a single instruction word	20	Operations register – memory (SRAM)
7	Skip two instruction words	21	Operations 16-bit register – memory (SRAM)
8	Skip the following instruction	22	24-bit and 32-bit operations
9	Call a subroutine	23	Load a register out of flash memory or SRAM
10	Return from a subroutine	24	Delays
11	Access memory and I/O with the Y- register as the base address register	25	Specialties
12	Access bytes in SRAM by indirect addressing (all registers r0...r31)	26	Save and restore register contents
13	Access words in SRAM by indirect addressing (registers A, B, C, X, Y, Z)	27	Supporting stack machine emulation
14	Clear a 16-bit register (A, B, C, X, Y, Z)	28	Subroutine call and parameter addressing

TABLE 2

The macros are divided into 28 functional groups: No. 1 to No. 28.

WE DO THE IMPOSSIBLE... EVERYDAY!

Our 5-Day Turn Includes
**Boards, Parts
AND Assembly.**



FREE LABOR

TRY SOMETHING DIFFERENT!

1st time customers receive FREE LABOR, up to \$1,000 on your first turn-key order.

OUR ASSEMBLIES START AT \$250

DOWNLOAD YOUR OFFER CODE HERE: Circuitcellar.com/SlingShot

We want to see **NEW DESIGNS** and **NEW CUSTOMERS!**

No more sacrificing quality for speed or price.

We are your **PCB ASSEMBLY SPECIALISTS!**

 **SlingShot**
ASSEMBLY



Find out why we're different at SlingShotAssembly.com/Different
Call for details: **720.778.2400** or Email: sales@sassembly.com

*Free labor, up to \$1000, for first-time customers on full turn-key assembly orders only.

©2019 COPYRIGHT SLINGSHOT ASSEMBLY

In the Xmega version of the AVR MCU, always access the low-order byte first. When writing into the data memory (SRAM), the byte order is not significant. Since AVR is a little-endian architecture, it is quite natural to access the low-order byte first.

When coding macro bodies, we have to consider the byte order of 16-bit accesses to peripheral units and the type of jump and call instructions the particular device support. This is indicated by definitions to be inserted at the begin of the main (application) program. For details see the documentation on the Internet.

If a macro parameter is a general address, it depends on the address range, which type of access instructions (memory or I/O) will be used. Furthermore, 16-bit accesses will be executed in the proper sequence (low-order or high-order byte first). If a macro parameter is a data memory (SRAM) address, all accesses will be executed by memory-related instructions, and the low-order byte will be accessed first.

REGISTER ADDRESSING

Register addresses must adhere to the assembly language conventions. The register names `r0` to `r31`, `x`, `y` and `z` are assembler keywords, the other have been declared by `.equ` statements. `xl`, `xh`, `yl`, `yh`, `zl`, `zh` are declared in the device definition (`.INC`) files. The particular registers of our virtual machine are declared in the macro source files. The assembler will properly insert register names passed

as parameters. However, the preprocessor will not accept register names in conditional statements nor support 16-bit registers made of two consecutive 8-bit registers, like `r1` and `r0`. As a consequence, we must be content with a somewhat cumbersome notation, or we have to declare separate macros for each register pair.

The first example is a macro to load a register pair with the content of an addressed word (of an I/O unit or out of the SRAM). The macro is called `GWLD`. It has three parameters, two registers, and the general address. The high-order and low-order byte need separate register parameters. Example:

```
GWLD r3, r2, cnt10
```

This macro will load the registers `r3` and `r2` with the content of a 16-bit count register within an Xmega MCU. A second example is a macro to add a 16-bit literal to the content of a register pair. You cannot declare a macro `addlit register_pair, literal`. To avoid the cumbersome syntax with two register parameters, we decided to declare individual macros for each register pair of our virtual machine. So, we have macros `addlita`, `addlitb` and so on. For example, `addlita 391` adds the value 391 to the content of the 16-bit register `A`.

However, there are alternative solutions. First, we could define 16 macros `gldw0`, `gldw2` and so on, each loading a register pair. Second, we could introduce new register names `r0w`,

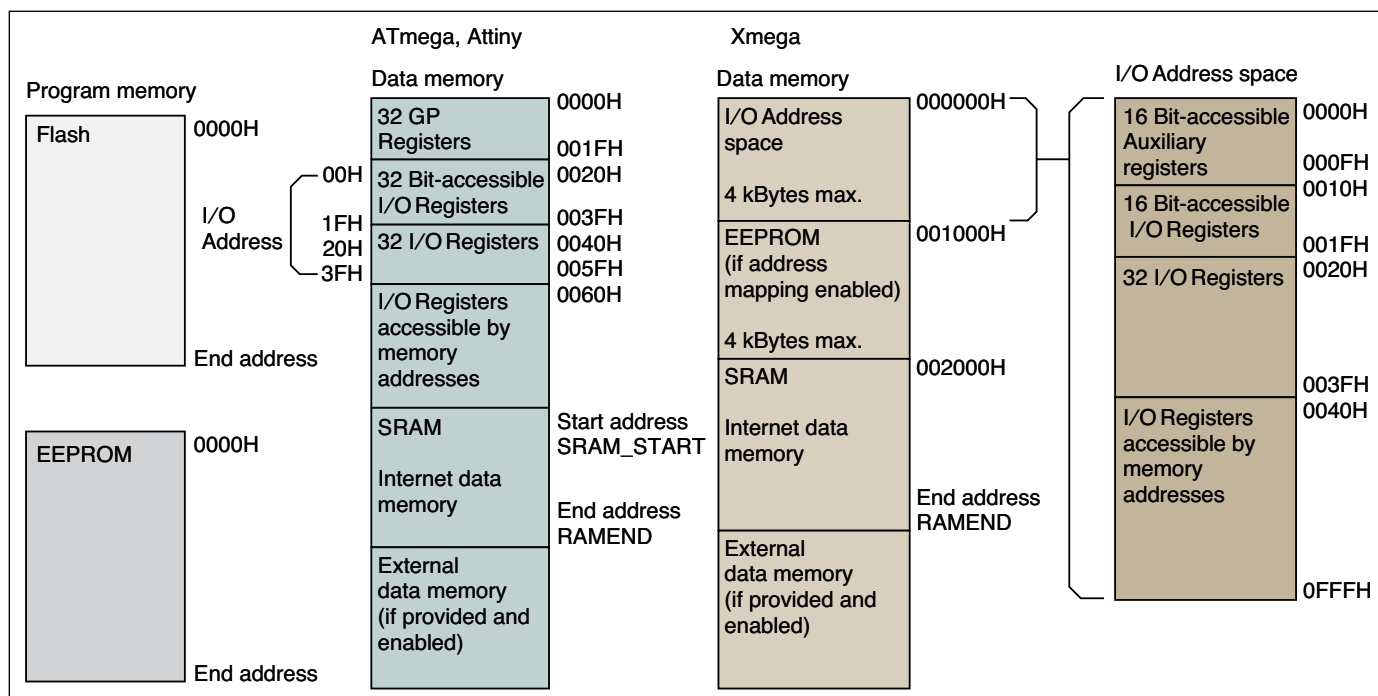


FIGURE 4

AVR memory and I/O addressing.

r2w and so on. These conventions would allow writing, for example, `gldw r0w, cnt10` or `addltw aw, 391`. For details, refer to the documentation on the Internet.

Let's look at some macro types:

Basic transports: Transport macros are basic move operations dealing with general addresses (**Table 3**). They keep care of the address space (data memory or I/O) and of the byte order (low-order or high-order byte first).

Single-bit operations: The bit is the most basic data structure. There are control bits and output signals to be set, and status bits or input signals to be sensed. Bit addressing should be eased in the registers, the peripheral units, and the data memory (SRAM). Macros are provided to set, clear or toggle a selected bit and to move a selected bit into one of the flags ZF or CF and vice versa. There are different ways to address the byte containing the bit (**Table 4** and **Table 5**).

Bit addressing in unified bit-fields (`unified_bitfield_adrs`):

When dealing with bits, the programmer must always know the address of the byte the particular bit belongs to. Let's say, example, we want to write routines supporting serial communication. While doing so, we come across a bit indicating the USART has transmitted a byte. When programming an ATmega16, we must know that this bit is called TXC and resides in the register UCSRA at bit position 6. When programming an ATXmega64A4U and using the USART 0 on port E, the bit is called TXCF and resides in the register USARTE0_STATUS at bit position 6.

As a remedy to such complexities, the concept of the unified bit-field address has been introduced. This type of addressing enables you to refer to individual bits, located anywhere in the general address space, by a single name, without having to worry about the byte address. The unified bit-field address is a general address extended by the bit address in the byte. To define such a bit, the general address is to be shifted one byte to the left (`address << 8`).

Mnemonic	Parameters	Function
gst	general_adrs, reg_adrs	Store a register content at a general address
gwst	general_adrs, reg_adrs, reg_adrs	Store the contents of two registers (r0...r31) as a 16-bit word at a general address
gld	reg_adrs, general_adrs	Load a register (r0...r31) with the byte at a general address
gwld	reg_adrs, reg_adrs, general_adrs	Load two registers (r0...r31) with the 16-bit word at a general address
glit	general_adrs, literal	Move an immediate 8-bit value (literal) to a general address
gwlit	general_adrs, 16-bit-literal	Move an immediate 16-bit value (literal) to a general address
gmov	general_adrs, general_adrs	Move the byte at the second general address to the first
gwmov	general_adrs, general_adrs	Move the 16-bit word at the second general address to the first
gclear	general_adrs	Clear the byte at a general address
gwcLEAR	general_adrs	Clear the 16-bit word at a general address

TABLE 3

Basic transports. This table also corresponds the layout of the tables in the documentation.

General mnemonic	Mnemonic in jump, skip, call and return instructions	Bit selection
bit	0, 1. Examples: <code>jp0</code> , <code>jp1</code>	A bit in a register r0...r31
gbit	g0, g1. Examples: <code>jpg0</code> , <code>jpg1</code>	A bit in a byte at a general address (data memory (SRAM) or I/O register)
ubit	u0, u1. Examples: <code>jpu0</code> , <code>jpu1</code>	A bit in an unified bit-field (see below)
dbitx	x0, x1. Examples: <code>jpgx0</code> , <code>jpgx1</code>	Byte address (SRAM) = <X> + displacement
dbity	y0, y1. Examples: <code>jpgy0</code> , <code>jpgy1</code>	Byte address (SRAM) = <Y> + displacement
dbitz	z0, z1. Examples: <code>jpgz0</code> , <code>jpgz1</code>	Byte address (SRAM) = <Z> + displacement

TABLE 4

Different ways to address a bit in a byte.

Mnemonic	Parameters	Function
gbit0	general_adrs, bit_adrs	Clear a bit at a general address
bit1	reg_adrs, bit_adrs	Set a bit in a register
ubit1	unified_bitfield_adrs	Set a bit at a unified bit-field address
dbitxt	displacement, bit_adrs	Toggle a bit. Byte address = <X> + displacement
bitz	reg_adrs, bit_adrs	Move ZF into a bit in a register. If ZF = 1, then bit = 0
dbityz	displacement, bit_adrs	Move ZF into a bit. Byte address = <X> + displacement. If ZF = 1, then bit = 0
ubitc	unified_bitfield_adrs	Move CF into a bit at a unified bit-field address
bit	reg_adrs, bit_adrs	Set ZF according to a bit in a register. If bit = 0, then ZF = 1
ubittoc	unified_bitfield_adrs	Set CF according to a bit at a unified bit-field address

TABLE 5

Some example macros to deal with single bits.

Here are some definition examples. The bits to be defined are called `sercom_txc`, `strobe` and `slave_buffer_empty`:

1) `sercom_txc` = bit 6 in the register UCSRA (for example, ATmega16):

```
.equ sercom_txc = (ucsr_a << 8) + 6
```

2) `sercom_txc` = bit 6 in the register USARTE0_STATUS (for example, ATXMega64A4U):

```
.equ sercom_txc = (usarte0_status << 8) + 6
```

3) `strobe` = bit 6 in port D:

```
.equ strobe = (portd << 8) + 6
```

4) `slave_buffer_empty` = bit 3 in the byte SERIAL_CHECKS (SRAM)

```
.equ slave_buffer_empty = (serial_checks << 8) + 3
```

For example, if you want to set the bit `slave_buffer_empty`, you simply write:

Mnemonic	Parameters	Function
jpnc	jump_adrs	Jump if CF cleared
jp0	register_adrs, bit_adrs, jump_adrs	Jump if bit in register cleared
jpu1	unified_bit-field_adrs, jump_adrs	Jump if bit at a unified bit-field address set
jpa	jump_adrs	Jump if above (unsigned)
jpge	jump_adrs	Jump if greater or equal (signed)
skipae	–	Skip one instruction word if above or equal (unsigned)
skipu02	unified_bit-field_adrs	Skip two instruction words if bit at a unified bit- field address cleared
skipfle	–	Skip next instruction if less or equal (signed)
callle	subroutine_adrs	Call subroutine if less or equal (signed)
retu0	unified_bit-field_adrs	Return from subroutine if bit at a unified bit- field address cleared

TABLE 6

Examples illustrating conditional jump, skip, call and return macros.

```
UBIT1 slave_buffer_empty
```

Without this provision, you would have to program something like:


```
LDS r16, slave_buffer_empty
ORI r16, 8 ; Bit 3 will be set in r16
STS slave_buffer_empty, r16
```

Macros of other functional groups support unified bit-fields, too. For example, to call a subroutine if the bit `slave_buffer_empty` is set, you simply write

```
CALLU1 slave_buffer_empty, slave_buffer_exception
```

Branching, subroutine call and return: The AVR's conditional branch instructions have only a 7-bit address field and, therefore, a short branch distance. The limits ($PC - 63$ to $PC + 64$) are related to the address of the current instruction. If the branch target is further away, you have to program around it, for example, by using a branch instruction to skip over an unconditional jump instruction. Appropriate macros support branching within the complete address space. Conditional execution has been provided by jump, skip, call and return macros (**Table 6**).

SUMMARY AND SUGGESTIONS

When you want to—or have to—program in assembler, and your primary goal is getting bulky application software up and running, a well-proven approach is to stay within the established ecosystem and to create some kind of runtime environment by writing appropriate macros and subroutines. It is wise to begin with a comparatively straightforward architecture and inexpensive hardware, like starter kits and small MCU modules [7]. To demonstrate the approach, we chose here to examine the AVR architecture. It goes without saying that the principles could be easily applied both to more advanced or to rather minimalist architectures and to 32-bit or 64-bit computing. 

Guiding the way to affordable autonomous systems
... with ACEINNA's high-accuracy OpenIMU™ solutions



IMU and RTK/INS solutions for:

- Commercial Drones
- Agriculture
- Construction
- Industrial
- Transportation



Link to www.aceinna.com/inertial-systems



Building a Smart Frying Pan

Connected Control for Chefs

There's almost no limit to what an MCU can be used for—including objects that previously had no electronics at all. In this article, learn how this Cornell University student built a Microchip PIC32 MCU-based system that wirelessly measures and controls the temperature of a pan on a stove. The system improves both the safety and reliability of cooking on the stove—and has potentially interesting commercial applications.

By *Joseph Dwyer*

I was the family chef for a number of years in my home, but since I began attending university, my mom has reluctantly taken over as house cook. Learning how to cook can be difficult, and my mom's tendency to multitask while food is on the stove causes a few small kitchen fires each year. My project, called Smart Pan, solves this problem, making cooking easier to learn, preventing overcooking and stopping fires.

The Smart Pan is an installable knob and handle set that controls your stove wirelessly. It consists of two separate components—a temperature sensing handle and self-correcting stove knob—that communicate

via radio (**Figure 1**). The installable handle detects the temperature of any pan and the knob turns to maintain a given temperature on any stove.

The Pan also connects to your phone, where myriad smart options can exist. From a smartphone, you can instruct the Pan to follow specific profiles, so you can cook just like your favorite chefs. The Pan contains simple rules to turn off if the temperature exceeds the smoke point of cooking oil. The knob and handle both use available 9 V batteries and are easily installable in the average kitchen. Both the handle and knob modules also use Sean Carroll's PIC32MC250F128B Small

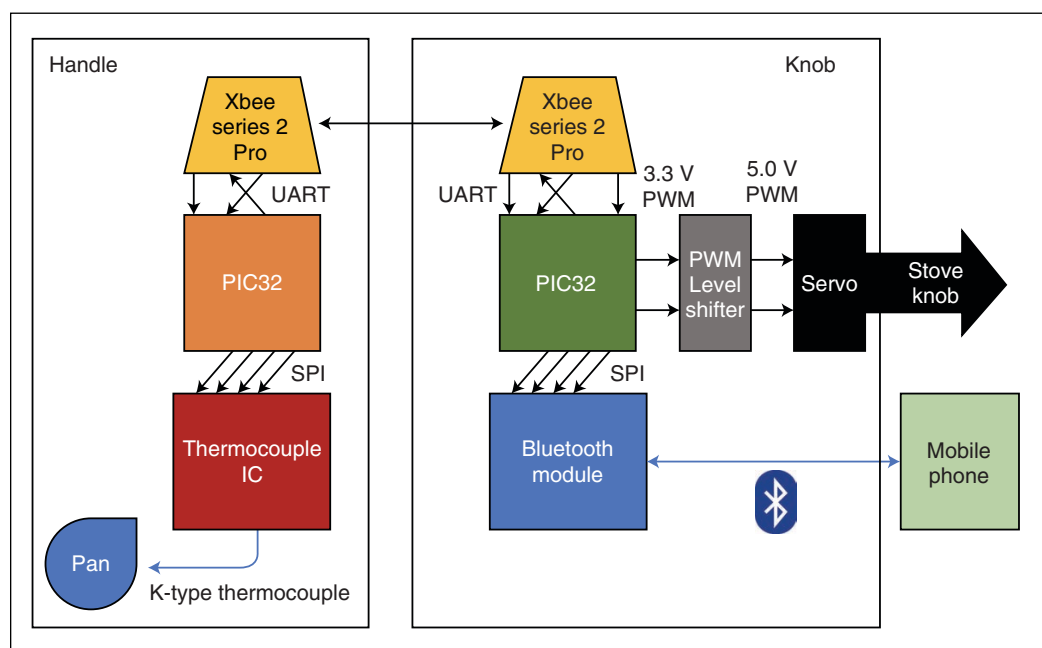


FIGURE 1

This system diagram shows the major electronic subsystems of the design.

Development Board. The *Circuit Cellar* article materials webpage has a link to the details of that board [1]. **Figure 2** shows a photo of with an overview of the project.

THE HANDLE

The handle mounts like a sleeve over most pan handles, and is secured in place with a pair of thumbscrews. I chose thumbscrews so that the handle could be slipped on and off different pans without any tools. The back thumbscrew prevents frictional load on the inside of the case when the pan is lifted. The front thumbscrew prevents the handle from slipping around inside the case.

The case is made of 3D-printed polylactic acid plastic, commonly referred to as PLA. The advantage of PLA is that it's readily available for prototyping. On the down side, it's quite susceptible to melting under heat. Luckily, most of the work of thermal isolation is done for us by pan manufacturers, to prevent customers from burning their hands. The thermocouple wire can provide only a limited cross-section for heat conduction, so these thermocouples should only allow a small throughput of heat.

PLA restricts the use case of the pan slightly, but incorporating additional cooking features is nearly impossible for additional reasons. For instance, when making a roast a chef will place the pan in the oven after browning the roast on the stovetop. PLA couldn't survive in an oven, but neither could any modern electronics. Robust microprocessors are only comfortable up to 80° C, so without a significant advance in processor materials or thermal efficiency, this device could never be used in the oven.

The ends of the proposed PLA handle contain small thermocouples attached to permanent magnets. Neodymium Iron Bromide (NIB) magnets were chosen both because of their availability and because the temperature at which their magnetic polarization breaks down is far outside the range of normal cooking temperatures.

These magnets attach to the sides of the metallic pan and conduct heat to the thermocouple. The magnets are separated from the thermocouples by Kapton tape. Kapton tape allows for complete thermal conductivity without any electrical conductivity. This prevents the electrically conductive magnets from interfering with the thermocouple measurement without also insulating the thermocouple from the heat that it needs to measure.

Based on information from this thermocouple, the knob computer will pick an appropriate amount to turn, and rotates the knob to keep a consistent temperature.



FIGURE 2
Project overview.

However, thermocouples tell us only the voltage across them, and the knob and pan computers are separate. Therefore, I measure the thermocouple with a thermocouple IC, and then transmit that information via the Microchip PIC32 microcontroller (MCU) and the XBee device. (**Figure 3**).

The Sparkfun MAX31855K is a breakout board with a thermocouple IC. It allows the PIC32 MCU to read the thermocouple via SPI. The module requires 3.3 V power, and draws a small enough current that the PIC32 board regulator can be used to power it. The IC provides 14-bit resolution over the entire range of K-Type temperatures. This gives us a resolution of 0.3°C, which is more than enough for accurate cooking. However, this temperature value must somehow be

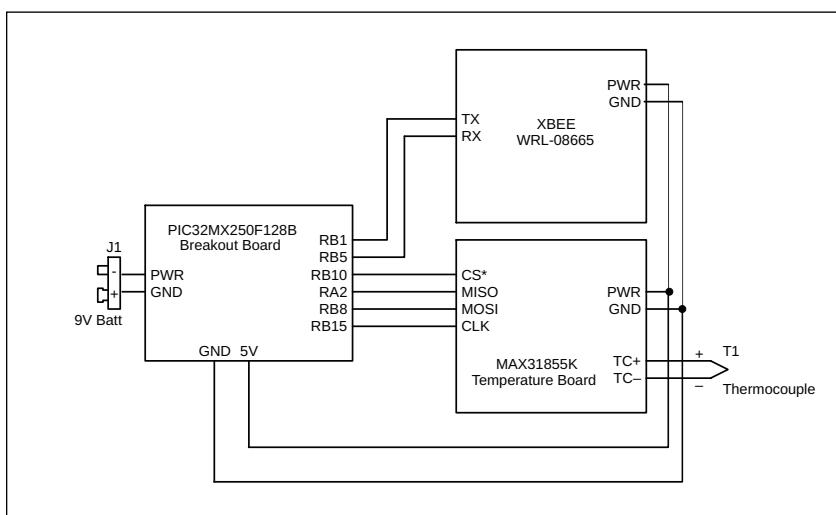


FIGURE 3
Pan handle hardware schematic

transmitted to the knob module, which we can do using the XBee module.

The XBee Series 1 module sends data from the handle module to the knob. XBees are hugely popular for a number of reasons, but I chose to use them in this project because they connect with simple serial ports. This allowed me to send the current temperature value without much hassle and without having to work with an unfamiliar protocol. The XBee board draws approximately 50 mA at 3.3 V, so it can be powered simply via the 3.3 V linear regulator on the PIC32. This increases the draw on the PIC32 linear regulator, but not so much that I see any problems.

These modules of course require a power source to operate. Both the handle and knob module use a 9 V battery as a power source. This is to ensure that a consumer could easily switch out batteries without much hassle. From the 9 V battery, power is downregulated to on-chip 3.3 V power for the PIC32's Bluetooth Module and Thermocouple IC, off-chip 3.3 V power for the XBees and off-chip 5 V power for the Servo Motor within the knob module.

THE KNOB

The knob module receives a transmitted temperature from the handle board, translates this temperature to a PID command and rotates a servo accordingly (**Figure 4**). The knob was designed to replace any standard range/stove/oven knob as easily as possible. In the current revision of the project, I'm using an old stove

knob. Conveniently, most stoves have the same notched control rod behind their knobs, so the knob can connect with most stoves by sliding on with a press fit. The battery is replaceable through a drawer in the front of the knob, so the knob doesn't need to be removed to change batteries. This 9 V battery supplies power to all the knob component modules, which are described below.

The knob receives commands through an XBee module identical to the one in the handle system. The commands are then translated in software on the PIC32 to a PWM command. The PIC32 then generates this PWM command and sends it on to a level shifter. This level shifter increases the magnitude of the commands and sends them to the servo motor. This process adjusts the knob such that the pan reaches the desired set temperature with a simple PID control loop.

The servo motor needs 5 V control logic to operate. However, the PIC32 Small Development Board is only configured to output 3.3 V logic. The level shifter answers this with a simple amplifier hack. A bipolar junction transistor (BJT) and two resistors can shift the peak value of the PWM from 3.3 V to 5 V.

A good way to think about how the level shifter circuit works is to notice that it is, in effect, just a common base amplifier of for the PWM. The base is held at +3.3 V when the input is high (also +3.3 V) and the transistor is off. This allows the collector to float up to the +5 V rail. When the input goes low, the transistor turns on hard and pulls the collector to just above the logic low level of the input. BJTs also have the advantage of fast switching times, so the level shifter introduces very little lag into the sense and respond control loop.

I also wanted the user to be able to control the temperature to which the knob sets. The Bluetooth module is from Adafruit, and is based on a Nordic Semiconductor nRF8001 chip. It connects to a mobile phone and has its own open source app that allows a user to send temperature set commands. This module then connects via SPI to the knob PIC32 and forwards those command values.

SOFTWARE OVERVIEW

To begin any sort of calculation, we need first to take the temperature from the thermocouple IC. We only need to read data from the thermocouple, so it's only necessary to connect to the MISO (Master In Slave Out) pin on the thermocouple breakout board. To read the MAX31855 over SPI, we pull the select pin (CS) to low, read the data then set the pin back to high. The value of the temperature that the MAX chip transmits is a 32-bit integer, where the last 14 bits represent the temperature in degrees Celsius. I found

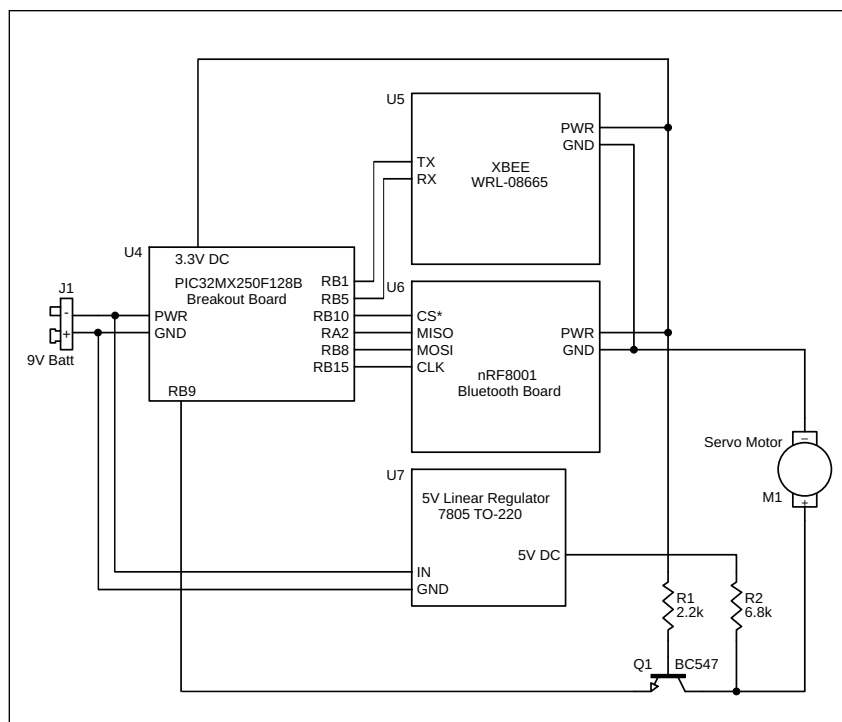


FIGURE 4

Knob hardware schematic

a tutorial for of the code in the MAX31855 website [2].

The pan handle pulls this information within the ISR (interrupt service routine), so that the data are consistently sampled, rather than sampled as the threads are scheduled. The precise timing of the samples may not seem important because we're severely oversampling relative to the problem, but the PID control loop is much more accurate with regular samples.

The pan handle's PIC32 converts that data into a string, so that it is accepted by the Zigbee software running on the XBee (**Figure 5**). I took the digit value of the temperature and sent it as a string. The XBees have a 100-byte UART buffer on board that ensures that the samples are transmitted when we're ready to receive them. The PIC32 in the knob currently sends a Zigbee receive command in the ISR, so that it's at least consistently timed on each individual board. The knob PIC32 receives this temperature value, compares it to a user set value and translates the disparity between the two values into the desired PWM value. Good control currently requires tuning for individual stoves.

After converting the temperature back to an integer that it can manipulate, the PIC32 also performs a sanity check on that temperature to ensure user safety. To prevent common house fires, the PIC32 has a shutdown protocol when the temperature reading exceeds the mean smoke point of several common cooking oils (150°C). If 20 sequential temperature readings are above this threshold, the set temperature is automatically changed from the user set value to room temperature. This immediately turns the pan off or to its lowest setting, and sends an alert to the user's phone. I also propose that a sanity timer be incorporated into these sanity checks to prevent a user from leaving the stove on at a high temperature.

The primary function of the PIC32, however, is to follow a given temperature profile. The knob PIC32 takes the user set temperature from the Bluetooth module over SPI much like the temperature board (Pulling CS to low, Reading the data, then raising CS again). I followed a tutorial from Adafruit's Bluetooth nRF8001 website that was fairly illustrative [3]. The output is also a 16-bit value, where 14 of the bits represent the temperature in degrees Celsius.

Conveniently, the Bluetooth module has a smartphone app that was easy to modify and get a minimum working product. The app uses Adafruit Bluefruit BT Connect on Android as a base [4]. It currently allows you to set variables stored in the Bluetooth module with their built-in GUI, so users can edit the USR_SET temperature value with a slider. The PIC32 accesses the Bluetooth chip in the same way that it accessed the MAX Thermocouple IC. This abstracts away the backend of Phone-to-Bluetooth communication in a way that makes it fairly easy to use.

The PIC32 then computes the error between the set temperature and the temperature read from the thermocouple. This error in temperature is the metric used in a PID control loop. I chose this linear error metric instead of something like mean square error, because I found that mean square error dramatically increased the "jitter" in the output.

With this error, I compute the PWM with a conventional PID control loop, which I tuned to my hotplate and pan at home. It's important to note that these tuning values will differ for different pans and stoves. In the final output, I average the previous 20 samples to prevent outlier measurements in the temperature data. This stops sudden swings in the PID calculation from causing large swings in output. This averaging is "hacked" together by writing several variables and averaging them.

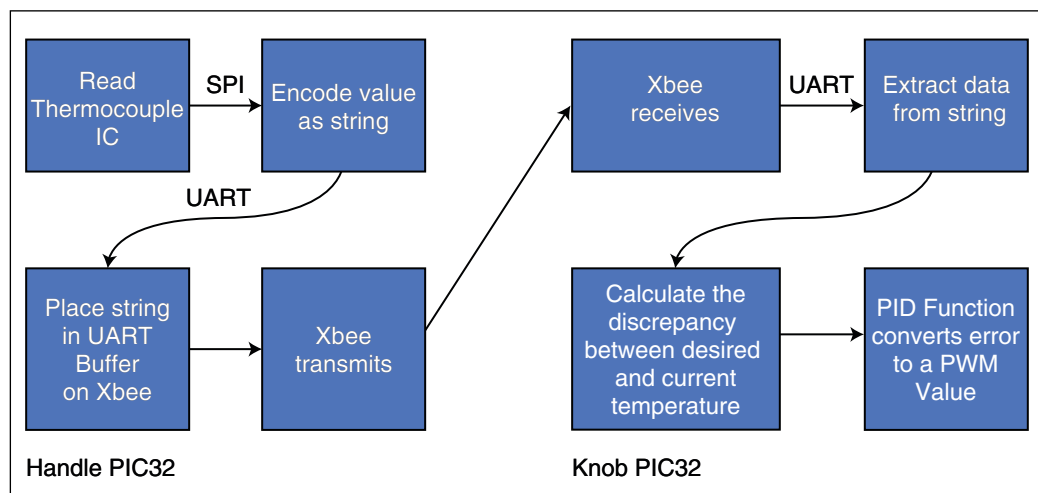
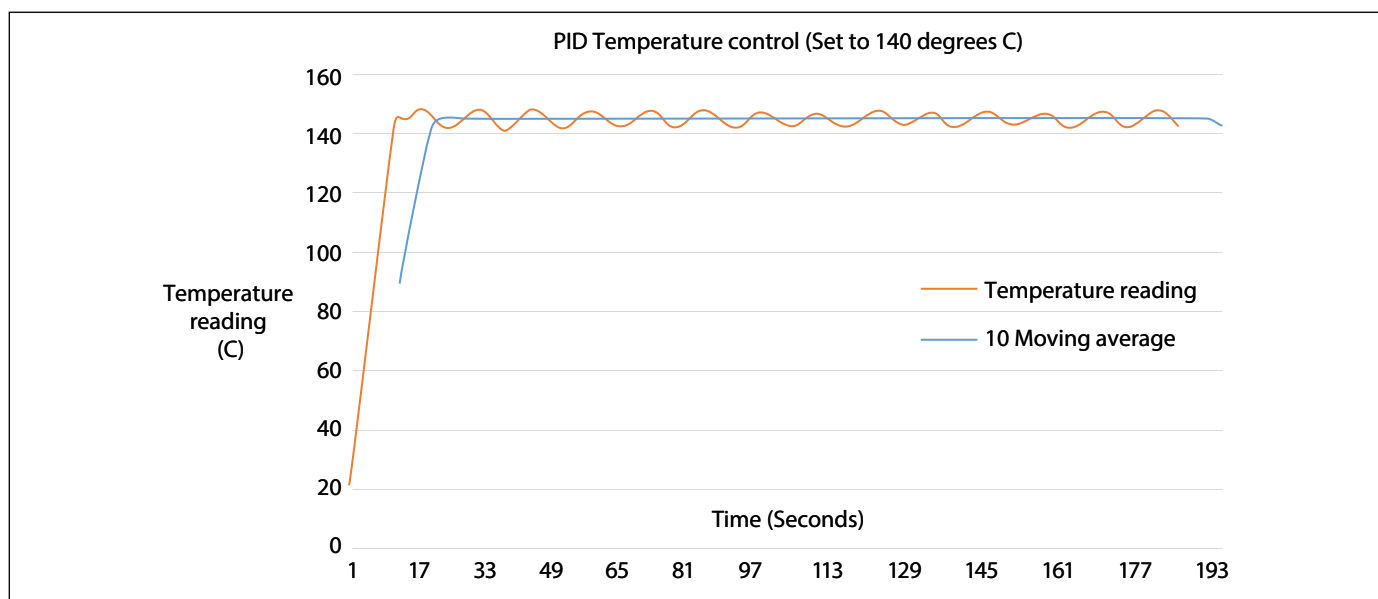


FIGURE 5

Shown here is the flow of one thermal data sample through the system.

**FIGURE 6**

PID dither and remedy using the moving average

While that may not be the best method, I did it that way because I know it works. The knob PIC32 then generates the appropriate PWM commands and sends them out over pin RB9.

TESTING AND IMPROVEMENTS

Although I had much initial success, one main issue kept arising during test. The latency time between a dramatic temperature change and the subsequent change in the pan. The pan has a significant thermal mass that takes several seconds to propagate to the outside. Making the system handle modular rather than drilling into the pan has this disadvantage. We can see the dither associated with a thermal “slew” in **Figure 6**. Using a second thermocouple allows us to predict the center temperature rather nicely with a heat diffusion model.

A better physical temperature model could improve this product significantly. The original design required drilling into the center of the pan and filling the cavity with thermal paste. This reduced latency to the speed of the electronics, but required a pan with a hole drilled into it for every unit. A preferable method would be to take two measurements on the edge of the pan and use a diffusion

model for the pan, to predict the temperature at the center from the two values at either side. I’ve tested this method by attaching thermocouples to the sides of the pan, but as of writing, it’s still unreliable.

The current revision of the project exists on a breakout board, but the breakout board should be unnecessary for a final product. Ideally, I’d route connections among all the components on a single handle board with the PIC32. This would give me a lot more control over the form factor of the final device, which could make it compact enough to make a long and narrow handle. The same huge improvement in form factor on the knob could be obtained by making another custom PCB.

I’d also change out the XBees for additional Bluetooth chips, so that I could get a less bulky, more reliable connection. XBee Radios are universal and reliable, but they have shortcomings. Radios are best for transmitting reasonably reliable data over large line-of-sight distances. However, in a kitchen setting, the distances are relatively short and an oven usually stands between the pan and knob. The original reason that I wanted to use the XBee was to familiarize myself with UART, but an improved design would perhaps incorporate a second Bluetooth board instead for more robust communication. My XBees also had antennas attached to them, but ultimately, I don’t think that they are necessary for communication over such short distances.

BOOSTING USABILITY

Several other improvements could improve the system’s usability. The user app currently doesn’t feature profiles or push notifications natively, but it could be significantly expanded to do so with a little more knowledge of app development. The user could be notified when

For detailed article references and additional resources go to:

www.circuitcellar.com/article-materials

References [1] through [4] as marked in the article can be found there

RESOURCES

Adafruit | www.adafruit.com

Microchip Technology | www.microchip.com

Nordic Semiconductor | www.nordicsemi.com


Sparkfun | www.sparkfun.com

their cook temperature exceeds certain preset values that are either dangerous or would ruin the food. The user could also load a whole cooking profile and see a small graph of the set temperatures over time with accompanying instructions for common meals. This is my ultimate vision for this product, and I'm hoping that other hobbyists can expand upon it and implement these features.

The knob controls temperature because a small servo rotates the rod behind the knob. An adhesive ring could be attached on the back panel of the case, to provide a counter torque for this knob to push off of. This solution has the advantage of being as universal as possible. The adhesive ring also has the advantage of connecting with any oven back panel. There are a few good options for the adhesive ring that might be better candidates for a final product. A ring of small magnets could hold the knob against the panel, but would only work on magnetic surfaces. However, it does have the advantage of being cleaner and reusable.

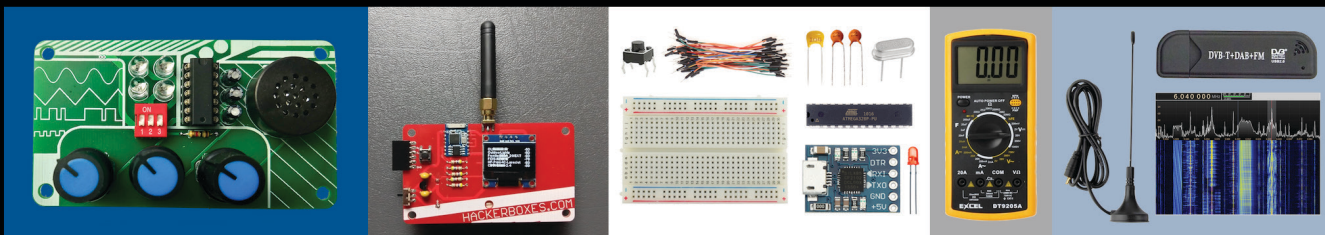
Ideally, I'd prefer to have users not need to tune their own knobs but I don't yet see a black-box style method to tune the knob automatically. PID auto tuning is the most significant barrier to ease of use. Currently, a user would have to tune the pan to his/

her own stove. This is a technical challenge for an inexperienced user that I would like to remedy, if possible. Auto tuning routines exist, but new open loops arise if I have to implement them.

This proposal has a lot of promise, but needs help in the implementation. I'd appreciate any additional comments and suggestions! Some other products have recently sprung up in this space, but they typically ignore the concept of modularity, making them expensive. Other products depend on other expensive pieces of technology. Inirv, a company that raised over \$175,000 on Kickstarter two years ago, built a system of smart knobs that costs \$299 and relies on an expensive smoke detector unit. That approach might be too little too late to avoid the beginnings of oil fires. In contrast, my Smart Pan implementation has a clear speed advantage here. 

ABOUT THE AUTHOR

Joe is currently a hardware engineer at SpaceX in Los Angeles, CA. He's interested in controls, machine vision and product development and maintains a lifelong passion for cooking. He can be reached at jmd456@cornell.edu.



HACKERBOXES

MONTHLY ELECTRONICS KITS

SUBSCRIBE NOW: HACKERBOXES.COM

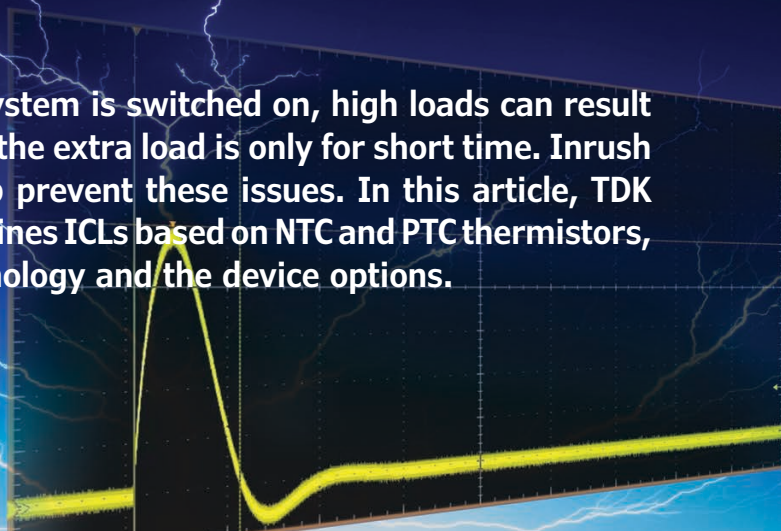


Inrush Current Limiters in Action

Circuit Guardians

At the moment a high-power system is switched on, high loads can result in serious damage—even when the extra load is only for short time. Inrush current limiters (ICLs) can help prevent these issues. In this article, TDK Electronics' Matt Reynolds examines ICLs based on NTC and PTC thermistors, discussing the underlying technology and the device options.

By **Matt Reynolds**,
TDK Electronics



When high-power devices and systems such as power supplies, frequency converters or on-board chargers are switched on, loads that are often many times the rated current can cause significant stress or damage. Although this extra load is only for a short period of time, it can damage the system, trip fuses or cause other issues with how the device operates. In order to protect the devices and circuitry, ceramic inrush current limiters (ICLs) may be used that are based on NTC and PTC thermistors (**Figure 1**).

With NTC (negative temperature coefficient) thermistors, the resistance decreases with increasing temperature. In PTC (positive temperature coefficient) thermistors, resistance increases as temperature increases. When a specific temperature is exceeded, PTC thermistors show a sharp rise in resistance.

High inrush currents come in two different types. First, inductive loads that occur in transformers and motors require very high currents to create the magnetic fields needed

to operate properly. Second, high-capacitance capacitors in DC links cause high charging currents and cause significant stress to the capacitors and especially to the rectifiers at the moment of connection (**Figure 2**).

The most traditional way to limit inrush currents is by using low-ohmic power resistor to reduce the current. However, once the inrush is over, the resistor continues to cause a power loss that affects the entire system, which is a significant drawback to this method.

Another, more effective method involves the use of thermistors as ICLs. Both NTC and PTC thermistors have thermal characteristics that can be used—although they differ in resistance characteristics. As a result, they may be used in different applications that require their different resistance characteristics. In some cases, they can be used in combination with each other to provide the desired resistance needed for the application. We will explore the characteristics and applications of both NTC and PTC thermistors below.

NTC THERMISTORS

NTC thermistors limit high input-side inrush currents. NTC thermistors are temperature-dependent resistors whose resistance drops as the temperature rises, and are typically made from ceramic materials. The resistance of NTC thermistors depends on the ambient temperature at the time it begins to receive power. When the ambient temperature is low, the NTC thermistor's resistance is relatively high resulting in longer charging times due to lower charging currents. However, higher temperatures cause the NTC ICL to be in a low ohmic state which limits its ability to suppress inrush currents. In other words, as more current flows through the component, it heats up and provides less resistance. In addition, losses of the rated current are also low.

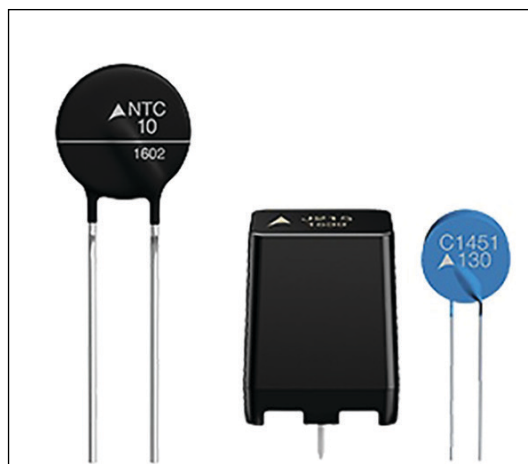


FIGURE 1

NTC inrush current limiter and PTC thermistors in housed and standard disk designs

When selecting an NTC thermistor, it's important to know the initial resistance and maximum current that will flow through the thermistor. The initial resistance must be high enough that when it is connected in series with the load, the current is limited and does not cause the fuse to trip or cause other damage to components including rectifiers. The maximum current is determined by the power rating of the load and as a result, the NTC thermistor must be de-rated.

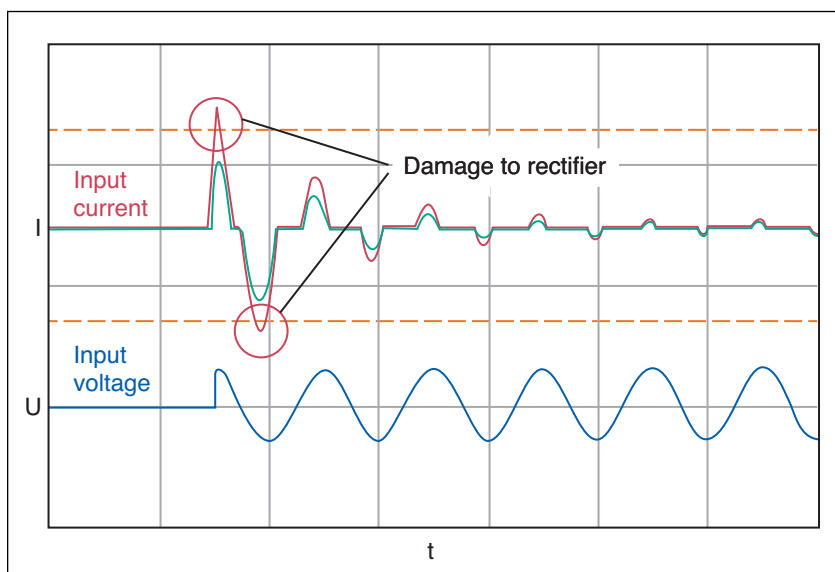
Cooling time of approximately 90 seconds depending on type should be ensured when using the ICLs. However, this can be problematic when loads are frequently switched at short intervals. The reason why is that a warmed-up NTC thermistor offers little current limiting and is very low ohmic. To overcome this, an NTC thermistor can be bypassed using a thyristor or relay just a short time after switching on and loads are at the rated current. When this occurs, the NTC thermistor does not experience ohmic-reducing warming.

A Zener diode and time constants determine the response time of the bypass circuit depending on the tolerances of the components (**Figure 3**). Due to the charging current the relay responds to the current requirement. If loads have high rated currents, the power demand of the circuit is less than the losses caused by the NTC thermistor's continuous current flow.

NTC thermistors have a high resistance at room temperature and when they are energized, they generate heat by themselves and the resistance falls as their temperature rises. Due to these attributes, NTC thermistors can be used as current protection devices for electrical and electronic devices that easily and effectively limit abnormal currents including an inrush current at the time of powering on. When used as current protection devices, NTC thermistors are also called power thermistors.

Although they provide fixed resistance, an NTC thermistor always causes a power loss and a decrease in performance. Therefore, an NTC thermistor limits an inrush current with its high initial resistance, and then its temperature rises because of energization and its resistance falls to a few percent of its level at room temperature. In that way it achieves a power loss that is lower than when a fixed resistor is used.

In other words, the effect of limiting inrush currents obtained by using an NTC thermistor is greater than that obtained by using a fixed resistor with comparable initial power losses. As a result of these characteristics, NTC thermistors can be used as inrush current limiting devices for switching power supplies, AC-DC power modules, DC-DC converters, industrial inverters and more.



PTC THERMISTORS

In DC link circuits, high-capacitance capacitor banks and capacitors may short circuit when switched on. PTC thermistors should be used instead of fixed resistors to have reliable current limitation. PTC thermistors offer more consistent and reliable protection against inrush current surges and short circuits, while providing accurate temperature control and measurement. When current flows through a PTC thermistor, it heats up and its resistance increases, making it self-protecting and providing a significant advantage to other forms of inrush current limiting.

This behavior is the opposite of NTC thermistors, making PTC thermistors intrinsically safe and limits the current to values that are harmless to the system in the case of a short circuit, something that fixed resistors cannot do. This is an ideal characteristic for frequency converters. In banks of capacitors, engineers should make sure not to exceed the maximum thermal capacitance and maximum permissible temperature of the PTC thermistors. The necessary thermal capacitance can be achieved by connecting the PTC ICLs in parallel.

FIGURE 2

Current flow in a rectifier with and without inrush current limiters (ICLs)

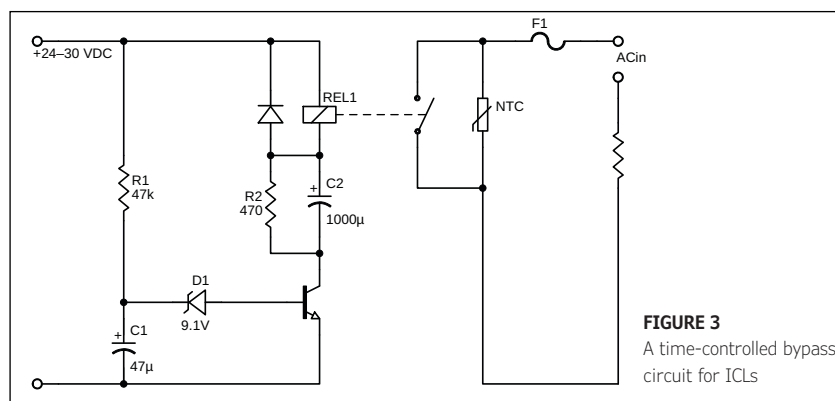
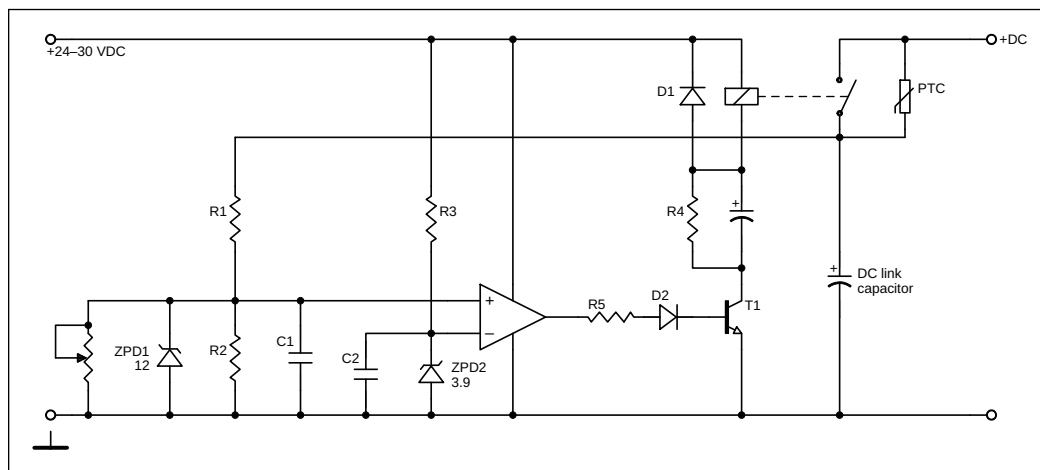


FIGURE 3

A time-controlled bypass circuit for ICLs

FIGURE 4

Voltage-controlled bypass for PTC thermistors



The PTC ICL must be bypassed after charging the DC link capacitors in normal operations to eliminate or reduce power losses. There must be no bypass, however, if there is a short circuit in the DC link—caused perhaps by damaged capacitors. The most significant parameter for a bypass circuit, therefore, is the DC link voltage (**Figure 4**). If it reaches the setpoint after charging it will not fault. However, it will short circuit if it remains at a very low value for a longer period of time. This enables a comparator circuit to be implemented with little effort, which bypasses the PTC thermistor only after charging of the DC link capacitor.

The inverting input of the comparator may be controlled by a Zener diode. When specified voltages are applied the comparator at the output trip to positive potential and switches the relay, causing the PTC thermistor to be bypassed. In this way, the varistor and the Zener diode serve to protect the non-inverting input of the comparator against overvoltages.


Because PTC thermistors are based on special semiconductor ceramics with a high positive temperature coefficient, they are

temperature-dependent. They exhibit relatively low resistance values at room temperature. When a current flows through a PTC thermistor the heat generated raises its temperature and once a pre-defined temperature—called a Curie temperature—is exceeded, the resistance of a PTC thermistor rises significantly.

This attribute can be used to protect circuits or devices from overcurrents. In this case, the overcurrent brings the PTC thermistor to a high temperature and the resulting high resistance then limits the overcurrent and eliminates this cause of possible malfunction. When the cause for possible failure is eliminated the PTC thermistor will cool down, and act as a resettable fuse and can trip. As a result, PTC thermistors can act as a robust overcurrent protection device for on-board chargers, industrial inverters, on-board DC motors, solenoids and more.

NTC AND PTC TOGETHER

It is possible to combine the advantages of NTC and PTC inrush current limiters and leverage both of their functions in the case of high-power loads that have DC link capacitances. The main applications for this are industrial power supplies and converters. When this is done, a voltage-controlled ON time should be employed to bypass the NTC thermistor on the power input side. To accomplish this, a relay with two changeover contacts is needed in which the NTC and PTC and switched at the same time.

Using a combined solution to address inrush current limitations results in better protection of components, reliably limiting current to prevent DC link short circuits and preventing the tripping of internal or supply-side fuses. Regardless of whether or not an NTC, PTC or both thermistors are employed, the entire system's high-power loads cause less damage and stress, thereby increasing the life of the design. 

ABOUT THE AUTHOR

Matt Reynolds is a director of marketing for piezo-electric and circuit protection devices at TDK Electronics. He has two decades of experience in products including NTC and PTC thermistor, disk and multilayer varistors, and various electronic ceramic components. He holds a Bachelor of Science degree in Ceramic Engineering from Alfred University (NY) and a Master of Engineering in Materials Science from University of Virginia. He has published and presented on technical topics, at industry events including APEC, and IEEE conferences such as ECCE.

For detailed article references and additional resources go to:
www.circuitcellar.com/article-materials

RESOURCES

TDK Electronics | www.tdk-electronics.tdk.com

Use your Pi to collect sensor data to the Google Cloud



U r s a L e o

Download our Raspbian package to turn your Pi into a Google cloud gateway.

Display data on dashboards, store and download it, use it to drive emails, texts and other alerts.

Create a cloud account and receive a free LED debug board for your Pi*

- LEDs indicate BLE, internet and cloud connectivity
- Console interface
- Safe power down switch

circuitcellar.com/ursaleo

*Offer available for North America and Europe only



Embedded Solutions Enable Smarter Railway Systems

Computing, Connectivity and Control

SPECIAL FEATURE

Railway systems keep getting more advanced. On both the control side and passenger entertainment side, embedded computers play critical roles. Railway systems need sophisticated networking, data collection and real-time control—all while meeting safety standards.

By **Jeff Child**,
Editor-in-Chief

There's no doubt that railway systems represent one of the most dynamic segments of embedded computing design.

There's a lot for embedded systems to do aboard trains—ensuring both safety and precise control for the train, but also for the increasingly sophisticated entertainment systems installed on today's modern trains.

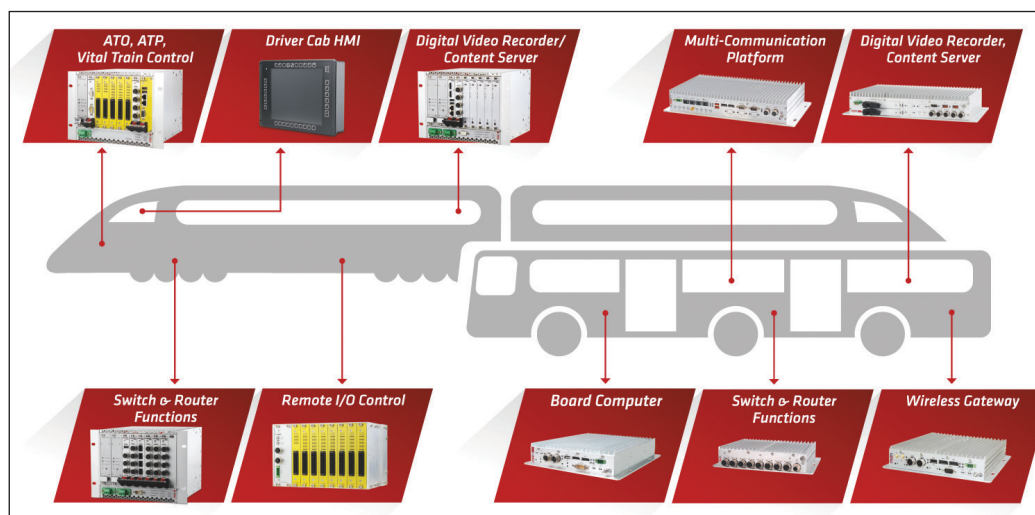
Meanwhile, trains are evolving into moving Internet-of-Things (IoT) platforms, as system developers strive to leverage the many benefits of data collection and passenger monitoring. Even embedded artificial intelligence (AI) is finding its way into the mindshare of railway system developers.

Exemplifying these trends, MEN Micro is a leading example of an embedded computer vendor deeply immersed in railway system technology development. Among its offerings

are its line of EN 50155, ISO 7637-2 and IRIS certified electronics that are embedded into trains (and buses) for control, supervision, communication, passenger information, security and testing. **Figure 1** shows an overview of the MEN's solutions along those lines. EN 50155 is one of a handful of standards targeted specifically for railway systems. **Table 1** shows a summary of these standards.

DIN-RAIL MOUNTING

In January, MEN Micro introduced the MC50M, its latest modular computer for DIN rail mounting. To clarify, "rail" in this context is referring not to railroad rails but rather to the metal DIN rail, a standard type of mounting used in industrial control equipment inside equipment racks. MEN's DIN rail concept is designed for flexible configuration of module combinations and is suitable for embedded

**FIGURE 1**

An overview EN 50155, ISO 7637-2 and IRIS certified electronics that are embedded into trains (and buses) for control, supervision, communication, passenger information, security and testing.

IoT applications in various markets. DIN rail mounting (35 mm) is standard. Wall and 19" rack mounting are possible using adaption brackets.

The EN 50155-compliant box is based on Intel's Atom E3900 series with low power dissipation and scalability in performance and memory. The modular expansion concept makes the DIN rail family a cost-effective and flexible solution. For memory, the system provides up to 8 GB of DDR3 SDRAM and an M.2 NVMe slot for mass storage. The box embeds a Trusted Platform Module for security and for I/O the MC50M provides Gbit Ethernet, USB 3.0, RS-232, R-S485/422 and DisplayPort. Input voltage is 24 VDC nominal with ignition and it supports a full-range of PSUs from 9 VDC to 60 VDC. Operating temperature is -55°C to +70°C.

According to MEN Micro, the MC50M is well suited for transportation functions such as

security gateways, predictive maintenance, CCTV, ticketing systems or as a diagnostic server. The MC50M can be used as a stand-alone product or in combination with a range of pre-fabricated extension modules, providing additional features and short delivery times.

Extension modules can provide application-specific functions such as wireless communication (LTE advanced, WLAN, GNSS), MVB, CAN bus or other I/Os. A removable storage shuttle supports the integration of one or two 2.5" SATA hard disks/SSDs. The wide range PSU allows isolated power supply from 24 VDC to 110 VDC nominal and extends the entire system to EN 50155 compliance.

The board management controller provides increased reliability and reduces downtime. The Trusted Platform Module supports security and encryption features. With an ignition switch for remote startup

EN 50155	This is an international standard required for all electrical equipment used on rolling stock for rail applications. EN50155 involves systems standing up to wide temperatures, shock and vibration, a wide range of voltage to power the systems, electromagnetic parameters and evidence of tested performance and reliability. Any EN50155 system is going to be rugged enough to take the sustained pressures of a rail environment.
EN 50121	This is specifically the electromagnetic compatibility of an electrical device within rolling stack. EN50121 has to be passed as part of the EN 50155 compliance. The standard restricts electromagnetic interference levels allowed to be emitted to the outside world as well as within the train vehicle itself. It measures any potential risk to signaling and telecommunications apparatus as well as fixed power supply installations.
EN 45545	This standard represents fire standards on railway applications with requirements for the behavior of materials and components.
EN 61373	Again, this standard is composite to the EN50155 for railway applications. The EN 61373 is concerned with shock and vibration, testing any electronic device over differing axis and timescales up to 5g.

TABLE 1

Shown here is a summary of the key certifications for embedded computers for railway systems. (Source: Assured Systems).



FIGURE 2

The TRACe-RM404-TR is a fanless railway computer specifically designed for train control and communication applications. Designed as a robust and compact 19", 1.5U box computer (compliant to EN60297-3-100), it can easily fit any existing railway equipment.

and shutdown control, the platform provides additional energy saving features. The aluminum housing with cooling fins ensures conductive cooling and fanless operation. The MC50M has no moving parts, so it can be operated maintenance-free. The long-term availability of 15 years from product launch minimizes life cycle management by making the MC50M available for at least that period.

SECURITY AND SAFETY

Security and safety go hand-in-hand when it comes to railway computing systems. With that in mind, in April Kontron and SYSGO jointly started the development of an integrated platform for safety-critical railway solutions based on Kontron's SAFe-VX hardware. Their aim was to provide system integrators with a solid and flexible basis for certifiable applications in trains and signaling.

Kontron's hardware is already used in many railway systems and has been certified up to SIL-4, the highest level of the IEC 61508 standard for functional safety of electronic systems. The SAFe-VX Kontron hardware platform will run under SYSGO's real-time operating system PikeOS, which is already used in EN 50128/SIL-4 certified systems, on both single and multi-core architectures. Additionally, PikeOS is the only real-time operating system with separation kernel certified to EAL 3+ Common Criteria security standard to fulfill functional safety as well as security requirements on the same system.

According to Kontron, several railway application developers are already using

PikeOS on the Kontron hardware. Kontron and SYSGO have identified several opportunities for autonomous train driving and rail signaling applications, where SAFe-VX vital processing platform is a good match. The common platform is expected to provide developers a solution that meets the most stringent functional safety and embedded security requirements.

CONTROL SYSTEM SOLUTION

On the control side, among Kontron's latest offerings is its TRACe-RM404-TR, a fanless railway computer announced in January. EN50155-certified, it is specifically designed for train control and communication applications (**Figure 2**). Designed as a robust and compact 19", 1.5U box computer (compliant to EN60297-3-100), it can easily fit any existing railway equipment. Kontron says the system has already been chosen for a train retrofit by a large rail system solutions provider in Asia, supporting the train control in an automated metro system.

The TRACe-RM404-TR's 10 year product lifetime combined with the long-term support services ensure long service life, up to 25 years and more. The first TRACe-RM404-TR variant features the Intel Atom x5-E3940 quad-core 1.6 GHz high performance-per-watt processor, with 2 GB DDR3L memory up to 1,866 MHz (optional up to 8GB DDR3L) and 64 GB Industrial MLC SSD memory.

The box computer provides multiple communication ports, making it well suited for train control applications. It features three independent Gbit Ethernet network ports (on M12 connectors), 2x RS-232/422/485 isolated ports with galvanic isolation, 8 isolated digital inputs and 8 isolated digital outputs for operation plus 2x USB, 1x GbE RJ45, 1x RS-232 and DisplayPort interfaces for maintenance purposes. It comes with an EN50155 class S2-C1 ultra-wide range power supply (from 24 VDC to 110 VDC nominal input voltage range) adapted to all types of railway vehicles from light rail vehicles to high-speed trains. Its rugged EN50155 fanless and low-power design ensures high performance and high reliability, operating within a temperature range of -25°C to +70°C.

The TRACe-RM404-TR product can accommodate several optional wireless (4G LTE, Wi-Fi) interfaces, field buses (CAN 2.0, MVB) and/or additional optional I/Os (Audio, USB) to match any other railway applications such as onboard CCTV, entertainment/

infotainment PIS or train-to-ground communications.

PoE SUPPORT

Power-over-Ethernet is catching on in a variety of applications, and railway systems are no exception. Axiomtek has leveraged the technology in its latest transportation-focused box-level system. In May, the company announced its tBOX400-510-FL, a two-in-one transportation-certified box PC with built-in layer 2 managed PoE switch for IP surveillance applications (**Figure 3**). The tBOX400-510-FL is certified with CE (Class A) and FCC and is in compliance with E-Mark, ISO 7637-2, EN 50155, EN 50121-3-2 and EN 45545-2, IEC 60945 and DNV 2.4.

This high-performance transportation box PC is powered by the 7th generation Intel Core (Kaby Lake) or Intel Celeron 3965U processors, along with dual DDR4 SO-DIMM slots for up to 32 GB of memory. The built-in 10-port managed switch is powered by Qualcomm's chip. With its high performance, industrial-grade design and full-strict certifications, the tBOX400-510-FL is suited for transportation-related applications such as onboard security surveillance and video management by managed switch function.

Axiomtek's tBOX400-510-FL has a built-in 10-port managed switch with 8-port 10/100 Mbps PoE and 2-port Gbit LANs, featuring VLAN, QoS and PoE scheduling for the more secure and smooth network. For reliable operation in severe environments, the rugged fanless transportation box computer

is designed to withstand a wide temperature range from -40°C to +60°C and vibration of up to 3 Grms. It also supports 24 VDC to 110 VDC for railway applications, and other voltage configurations for other applications.

The all-in-one tBOX400-510-FL has two M12 A-coded Gigabit LANs and eight M12-type D-coded PoE ports with a total power budget of 120 W in the rear panel. Four USB 3.0 ports, two audio ports, one Gbit LAN with Intel i210-IT, one DVI-I and one RS-232/422/485 are in front of the system, while isolated DIO is reserved for request.

The IP30-rated embedded system supports two swappable 2.5" SATA HDD drives with Intel RAID 0/1 function. There are three full-size PCI Express Mini Card and two SIM card slots for 3G/4G, GPS, Wi-Fi and Bluetooth connections. The DC power input is available in M12 connector for railway applications or terminal block for marine and vehicle applications. One PoE power input is available for external PoE power supply.

IN-VEHICLE FOCUS

Also providing a PoE-based solution, Neusys Technology has its Nuvo-7100VTC Series listed as "Coming Soon" on its website. The system is an Intel 8th-Gen Core i7/i5/i3 in-vehicle fanless embedded computer that is EN50155 certified and provides 4x M12 PoE+ ports, DIO, CAN bus and RAID. With 64 GB DDR4 memory, and Intel's 8th-Gen Core processors with up to 6-core/ 12-thread architecture, the system offers a significant performance increase over previous generation Neusys

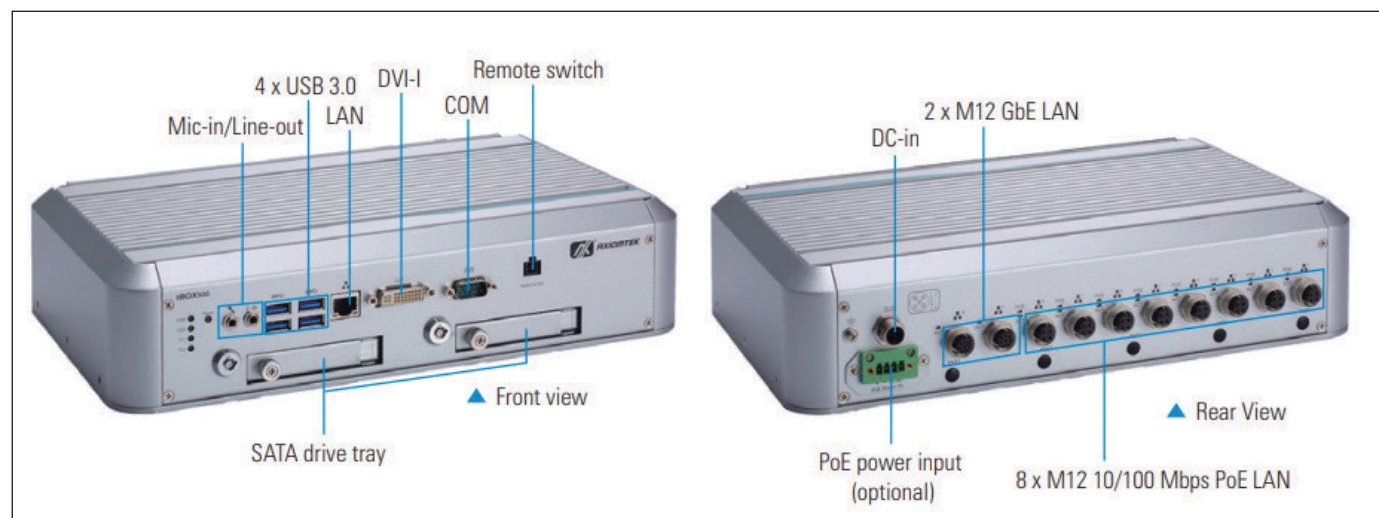


FIGURE 3

The tBOX400-510-F is a two-in-one transportation-certified box PC with built-in layer 2 managed PoE switch for IP surveillance applications. The tBOX400-510-FL is certified with CE (Class A) and FCC and is in compliance with E-Mark, ISO 7637-2, EN 50155, EN 50121-3-2 and EN 45545-2, IEC 60945 and DNV 2.4.

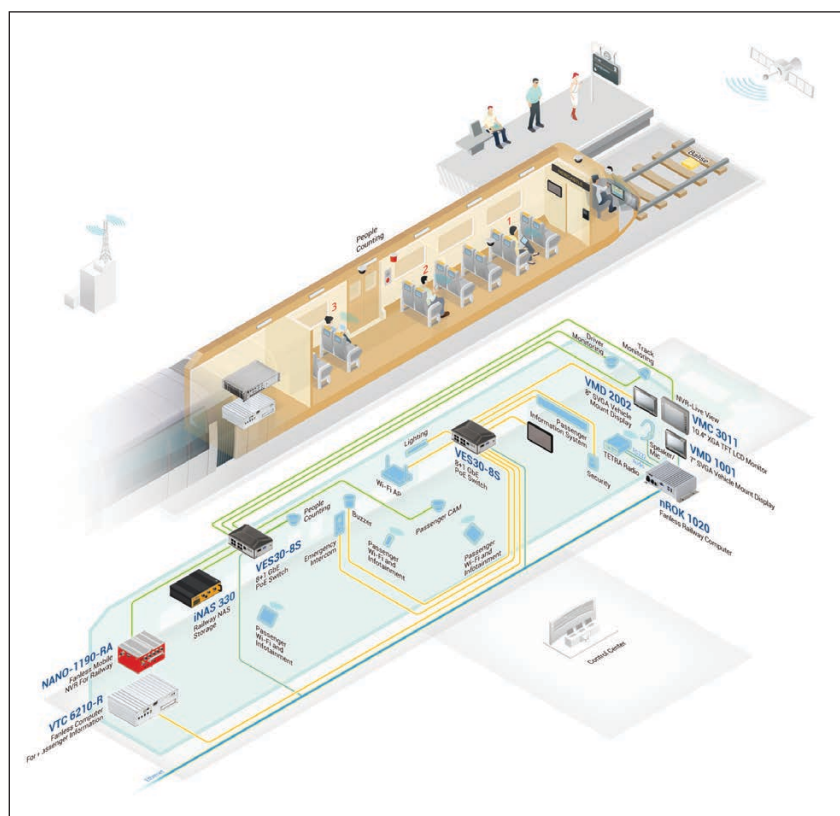


FIGURE 4

Illustrated here is NEXCOM's depiction of a Smart Public Transit railway implementation. With GNSS, wireless data communication and computerized processing of sensor-generated data, railway telematics can collect, process and share vital information such as positioning, vehicle health and railway line data.

solutions for versatile in-vehicle applications.

The Nuvo-7100VTC offers four or eight 802.3at PoE+ ports to supply 25 W power to connected devices such as IP cameras with M12 (x-coded connectors) and connector screw-lock mechanisms on computer I/Os like Gbit Ethernet, USB3.0 and USB3.1 to guarantee extreme rugged connectivity in shock/ vibration environments.

Wireless connectivity is essential for modern day in-vehicle applications. System developers can simultaneously utilize two M.2 and three mini-PCIe sockets with corresponding wireless modules for 3G/ 4G, WIFI, GPS and CAN module for communication. Additionally, there is a 4G cellular module option that is certified to work with renowned US telecommunications company which can save developers implementation time and cost. Nuvo-7100VTC also features isolated CAN bus for in-vehicle communication, isolated DIO for sensor/ actuator control, 8~35 V wide-range DC input with ignition power control and is in compliance with E-Mark and EN 50155.

SMART PUBLIC TRANSIT

For its part, NEXCOM sees its family of railway-focused electronic systems as key pieces within Smart Public Transit railway implementations—as depicted in **Figure 4**. With global navigation satellite systems (GNSS), wireless data communication and computerized processing of sensor-generated data, railway telematics can collect, process and share vital information such as positioning, vehicle health and railway line data. Operators can leverage that data as a tool to accurately track rolling stock positions, identify traffic events and measure railway performance to improve the safety and efficiency of the entire railway operation.

As an example, part of NEXCOM's offerings for such applications is its nROK 1020-A system. The box-level system is equipped with vehicle data acquisition, GPS and wireless communication and data processing capabilities. By bridging the gap between on-duty vehicles and central dispatchers, these vehicle terminals can help fleet, bus, and rail transport introduce telematics applications such as in-transit monitoring, usage-based insurance, preventive maintenance and more.

The EN50155-compliant nROK 1020-A addresses the special needs of rail transport. As a vehicle terminal, the nROK 1020-A can monitor passenger cars for doors being ajar, attempts to open emergency exits and



FIGURE 5

The EN50155 certified PIS-5500 is an AIoT (AI plus IoT) platform powered by an Intel Core i7 processor and integrated NVIDIA Quadro GPGPU module. The system is ruggedized for both wayside and onboard deployment with its wide range DC input and isolated I/O design.

rising temperature. To allow actions to be taken, the nROK 1020-A can communicate the unusual conditions to an operator's console over a closed-loop train network or report to a control center over 3G/LTE networks. At the same time the nROK 1020-A can run passenger signage applications, showing passenger information on two displays.

AI-IoT PLATFORM

Like IoT, AI is finding its way into a wide variety of embedded computing applications these days. And, according to ADLINK Technology, the rail industry is no exception. Fueled by intelligence from AI-driven systems and applications, railway operations are becoming safer, smarter and more reliable, significantly enhancing passenger travel experience and freight logistics services. These AI-driven applications only function with proper data input that is collected by massive numbers of IoT devices installed in stations, on trains and along tracks.

The company says that a successful implementation of such rail applications requires a seamless integration of AI and IoT technologies. Serving those needs, in May ADLINK announced its EN50155 certified PIS-5500 "AIoT" (AI plus IoT) platform (**Figure 5**). Powered by an Intel Core i7 processor and integrated NVIDIA Quadro GPGPU module, the system is ruggedized for both wayside and onboard deployment with its wide range DC input and isolated I/O design. It is well suited as an edge solution for real-time video/graphic analysis applications that are vital to today's increasingly complex railroad operations. To meet varying application requirements, the PIS-5500 is also available in variants featuring an additional two USB 2.0 via M12 connectors and two 2.5" SATA 6 Gb/s drive bays, as well as a version supporting +12 VDC power input only.

The target applications include—but are not limited to—passenger information systems, railroad intrusion detection, train station surveillance, onboard video security and railroad hazard detection. According to ADLINK, the PIS-5500 is being tested and deployed commercially by leading rail system integrators worldwide. In one application, the intelligent platform is installed on special rail inspection trains to process captured images of key wayside equipment in real-time.

With a sophisticated algorithm driven by parallel computing and deep learning, the application can effectively identify potential

equipment faults at a train speed of 120 km/h, and raise the alarm to notify maintenance crews. In another application, the PIS-5500 is used in a train station control office to analyze the real-time video stream received from the platform, says ADLINK. The application is able to not only detect suspicious behaviors and trigger alerts, but also conduct post-event analysis.

TOUCH PANEL PCs FOR TRAINS

Several vendors that make embedded computing systems for trains, also offer purpose-built touchscreen panel computers specifically designed for trains. Within the past several months, two such examples are Axiomtek's P6125, a 12.1" EN 50155-compliant touchscreen display and ADLINK's DMI-1210 driver machine interface (DMI) touch panel computer.

Another more recent example is Advantech's ITA-8000 series of fanless touch panel computers with a DMI. Providing real-time information and a touch panel design, locomotive operators can enjoy improved safety control and optimized decision making based on the collection and analysis of all relevant data (**Figure 6**). The ITA-8000 series is EN50155 certified, compliant with the EN45545 standard for railway operations and performs under the harshest of temperatures and environmental conditions.

Mounted in the locomotive and acting as "the brain" of the train infrastructure, the ITA-8000 series is a fanless, industrial-grade touch panel PC utilizing the latest Intel Atom x7-E3950 quad-core processor. It allows



FIGURE 6

The ITA-8000 series of fanless touch panel computers with a DMI provides real-time information and a touch panel design, so locomotive operators can enjoy improved safety control and optimized decision making based on the collection and analysis of all relevant data.

FIGURE 7

The MPT-3000RP is an EN50155 and EN45545 compliant transportation computer optimized for railway applications. Protected from water and other ingress per IP67, the 270 mm x 210 mm x 63 mm MPT-3000RP is equipped with a quad-core Intel Atom E3845 clocked at 1.91 GHz.



drivers to seamlessly monitor and adjust settings based on platform and carriage conditions from the driver's cab. This DMI product is designed for easy operation and displays a vast range of real-time information. Lockable front hotkeys seamlessly link the driver to the rolling stock control system.

The ITA-8000 series offers wide input power range—from 24 VDC to 110 VDC—that fulfills the diverse needs of different manufacturers. In order to further increase flexibility, the ITA-8100 series reserves a PCIe slot and a USB 3.0 slot on board. By adding a different optional module, the ITA-8000 series supports a variety of transportation communication protocols, such as MVB, and CAN bus.

A customized interface can also be implemented upon request. The series

features two panel sizes with 1024 x 768 display resolution and highly sensitivity capacitive touch: 10.4" for the ITA-8100 and 12.1" for the ITA-8120. The high-brightness anti-glare LCD panels ensure readability in both dark environments and under intensive sunlight. Both the ITA-8100 and ITA-8120 offer the option of UIC612-01-compliant keypads for control or display purposes.

The ITA-8000 series is specially designed for performance under tough of conditions, meeting the EN50155 and IEC61373 certification for rolling stocks applications and adhering to the strictest safety standards. The series can operate under temperatures ranging from -40°C to 70°C (class TX), and $\pm 40\%$ of selected input voltage. It can tolerate 20 ms interruption (class S3) and 30 ms supply break (class C2). The front panel is IP65-rated for maximum protection.

MULTIPURPOSE SYSTEM

Ibase Technology makes several products aimed at railway systems, some targeted at train-based digital signage. Last October, the company announced that its new Atom E3845 based MPT-3000RP—an EN50155-TX certified, IP67-protected railway PC—won a Taiwan Excellence Award along with a "DRD-037PC" railway signage PC and two recent Ibase embedded systems.

The MPT-3000RP is an EN50155 and EN45545 compliant transportation computer optimized for railway applications (**Figure 7**). However, instead of acting as a signage player, this is a more general-purpose computer designed to control a variety of onboard systems. Protected from water and other ingress per IP67, the 270 mm x 210 mm x 63 mm MPT-3000RP is equipped with a quad-core Intel Atom E3845 clocked at 1.91 GHz. No OS support is listed.

Users can load up to 4 GB DDR3L-1333 via dual slots, and store data with a CFAST slot and a 2.5" SATA bay. In addition, one of the three M.2 sockets is designed for 2280-form factor SATA cards. A second M.2 is designed for wireless 3042 cards, and a third M.2 slot is for a variety of 2230 cards (USB 2.0/PCIe). The system also provides half- and full-size mini-PCIe slots with USB 2.0 and USB 2.0/PCIe support, respectively. Dual SIM card slots and 4x antenna connectors are also available.

All the external interfaces use rugged M12 connectors. Provided are 2x GbE, 2x USB 2.0 and 3x serial ports, as well as single VGA, CAN

RESOURCES

ADLINK Technology | www.adlinktech.com

Advantech | www.advantech.com

Axiomtek | us.axiomtek.com

Cincoze | www.cincoze.com

Ibase Technology | www.ibase.com.tw

Kontron | www.kontron.com

MEN Micro | www.menmicro.com

Neosys Technology | www.neosys-tech.com

SYSGO | www.sysgo.com

and wide-range voltage GPIO ports. There's also an M12 connector for the "DC 9 V ~ 36 V / 72 V / 110 V" input. The company says, the system provides "interchangeable modular power supplies supporting a variety of inputs" and "full vehicle battery power control." The MPT-3000RP OS is wall-mountable and supports -40°C to 70°C operation with an SSD. It also offers EN61373 compliant shock and vibration resistance. Other certifications include EN50153:2014, EN50121-3-2:2015, CE Class A and FCC Class A.

PCI/PCIe EXPANSION

Like many embedded systems, railway systems must be designed with upgrades in mind. Systems that can easily add new features or performance with modular expansion form factors can really help stakeholders save costs over the long run. Along just such lines, last October Cincoze introduced its latest rugged embedded computer, the DS-1200 series with PCI/PCI Express expansion.

The DS-1200 system is equipped with the Intel Q370 chipset and supports the 8th Gen Intel Core/Pentium/Celeron 35 W / 65 W LGA 1151 processors (**Figure 8**). The new 8th generation Intel Core processor (Coffee Lake) for the first time offers up to 6 cores, 12 threads and 1.4 times computing performance improvement comparing to the previous generation (Kaby Lake). The integrated Intel UHD Graphics can drive up to 3 independent display outputs and support 4K UHD resolution. DS-1200 series also supports DDR4-2666 SO-DIMM memory, USB 3.1 (Gen2) ports and ultra-fast PCIe x4 NVMe SSD.

DS-1200 series comes with rich I/O connectors, including 2x GbE ports, 8x USB ports, 1x DVI-I, 2x DisplayPort, 1x PS/2, 1x Line-out, 1x Mic-in, 2x RS-232/422/485 ports and a remote power/reset connector. The system also has 2x front accessible SIM card slots for redundant 3G/4G connections. In addition, DS-1200 series provides friendly features, including instant reboot, replaceable fuse and integrated SuperCap for easy maintenance.

The system allows users to expand I/O and functionalities through ready-to-use modules, such as GbE/PoE ports, serial ports, optical isolated digital I/O and power ignition function. Additionally, the system provides 3x full-size Mini PCI Express slots and up to 2x PCI/PCIe expansion slots for more I/O expansions. DS-1200 series is available in three different models: DS-1200 without




FIGURE 8

The DS-1200 system is equipped with the Intel Q370 chipset and supports the 8th Gen Intel Core / Pentium / Celeron 35 W / 65 W LGA 1151 processors. The system provides 3x full-size Mini PCI Express slots and up to 2x PCI/PCIe expansion slots for more I/O expansions.

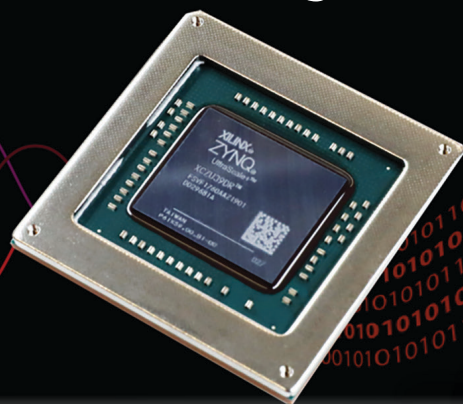
PCI(e) expansion slot, DS-1201 with one PCI(e) expansion slot, and DS-1202 with two PCI(e) expansion slots. By choosing optional riser cards, users can install versatile add-on cards for their specific applications.

DS-1200 features fanless/cableless design, wide operating temperature (-40°C to 70°C), wide range DC power input (from 9 V to 48 V), high tolerance of vibration/shock (5G/50G), and industrial-grade protections (OVP, OCP, ESD Surge and so on). The DS-1200 has passed various rigorous tests, as well as EN50155 (EN50121-3-2) and EN60950-1 certifications for operations in harsh environments.

It's clear that railway systems have become "power users" of the latest and greatest embedded computing technologies. To keep pace, the embedded computing vendors mentioned in this article continue to evolve their purpose-built, box-level systems to meet the demands of today's advanced railroad system implementations, both for on-board trains and for the infrastructure surrounding railroads. 

FPGAs Flex Their DSP Muscles

Pros at Signal Processing



Because they marry the combined benefits of powerful signal processing and system-level integration, FPGAs now rank as a key technology for embedded system developers. FPGA vendors are keeping pace with both chip- and IP-level solutions that meet today's system design demands.

By **Jeff Child**,
Editor-in-Chief

Today's FPGAs provide the kind of system-oriented digital signal processing (DSP) requirements in demand across a variety of applications—including broadcast video, financial processing systems, machine learning, software-defined radio and many others. Meanwhile, it is already a given these days that FPGAs have become complete systems-on-chips (SoCs).

While the trend toward FPGAs with general-purpose CPU cores embedded on them is nothing new, the latest crop FPGA architectures have moved toward supporting artificial intelligence (AI) and machine learning types of processing. Even within the past six months, FPGA vendors have announced new solutions that improve upon these processing levels. Aside from CPU and DSP processing, another big advantage of FPGAs lies in their ample, programmable, high-speed I/O, which is why they are often found close to the analog-to-digital converters (ADC) in radio frequency (RF) and radar applications.

6 GHz SPECTRUM SUPPORT

Exemplifying those trends, Xilinx in February announced an upgrade to its Zynq UltraScale+ RF SoC) portfolio adding greater RF performance and scalability. The new generation of these devices can cover the entire sub 6 GHz spectrum, which is a critical need for next-generation 5G deployment, says Xilinx. They support direct RF sampling of up to 5 GSps, 14-bit ADCs and 10 GS/s 14-bit digital-to-analog converters (DACs), both up to 6 GHz of analog bandwidth.

The RFSoC portfolio now includes the Xilinx Zynq

UltraScale+ RFSoC Gen 2 and Zynq UltraScale+ RFSoC Gen 3. Gen 2 is now in production and meets regional deployment timelines in Asia and supports 5G New Radio. The Gen 3 device provides full sub-6 GHz direct-RF support, extended millimeter wave interface and up to 20% power reduction in the RF data converter subsystem compared to the base portfolio (**Figure 1**). The product will be available in 2H 2019. Thanks to pin-compatibility across the portfolio, system developers can design and deploy their systems now using first-generation devices with a roadmap to Gen 2 and Gen 3 for greater performance.

The new products monolithically integrate higher-performance RF data converters that deliver the broad-spectrum coverage required for the deployment of 5G wireless communications systems, cable access, advanced phased-array radar solutions and additional applications including test and measurement and satellite communications. By eliminating discrete components, the devices enable up to a 50% power and footprint reduction, making them well suited for the needs of telecommunications operators seeking to enable massive multiple-input, multiple-output base stations for their 5G systems, according to Xilinx.

DATA-CENTRIC APPROACH

With an eye toward solving challenges in data-centric systems, Intel's Programmable Solutions Group (PSG) in April announced Agilex, a new family of FPGAs designed to enable customized solutions that address the unique data-centric challenges across embedded, network and data center markets.

The Intel Agilex family combines an FPGA fabric built on Intel's 10 nm process with heterogeneous 3D SiP technology. This provides the capability to integrate analog, memory, custom computing, custom I/O and Intel eASIC device tiles into a single package with the FPGA fabric. Intel provides a custom logic continuum with reusable IP (Intellectual Property) through a migration path from FPGA to structured ASIC. One API provides a software-friendly heterogeneous programming environment, enabling software developers to easily access the benefits of FPGA for acceleration.

According to Intel PSG, system developers need solutions that can aggregate and process increasing amounts of data traffic to enable transformative applications in emerging, data-driven industries like edge computing, networking and cloud computing. This includes edge analytics for low-latency processing, virtualized network functions to improve performance and data center acceleration for greater efficiency.

The Intel Agilex FPGA is the first FPGA to support Compute Express Link, a cache and memory coherent interconnect to future Intel Xeon Scalable processors. Agilex's 2nd-gen HyperFlex architecture provides up to 40% higher performance or up to 40% lower total power compared with Intel Stratix 10 FPGAs. The Agilex supports PCI Express Gen 5, offering higher bandwidth compared with PCIe Gen 4. It also supports transceiver data rates of 112 Gbps. Advanced memory is provided via DDR5, HBM, Intel Optane DC persistent memory.

Intel also claims that the device is the only FPGA supporting hardened BFLOAT16 and up to 40 Teraflops of DSP. Each Intel Agilex DSP block can perform two FP16 floating-point operations (FLOPs) per clock cycle (**Figure 2**). Total FLOPs for FP16 configuration is derived by multiplying 2x the maximum number of DSP blocks to be offered in a single Intel Agilex FPGA by the maximum clock frequency that will be specified for that block.

NEURAL NETS AND IoT

Last year Lattice Semiconductor unveiled its Lattice sensAI solution, a technology stack that combines modular hardware kits, neural network IP cores, software tools, reference designs and custom design services. In May of this year, the company followed that up with major performance and design flow enhancements for the Lattice sensAI solutions stack.

The new enhancements to the Lattice sensAI solution stack include a 10x performance boost over previous version. This performance boost is driven by an updated Convolutional Neural Network (CNN) IP and neural network compiler with features like 8-bit activation quantization, smart layer merging and a dual-DSP engine.

The new version expands neural network and machine learning frameworks support including Keras. It also provides support for quantization. Fraction setting schemes for neural network training eliminate iterative post-processing. Simple neural network debugging can be done via USB. New customizable reference designs in Lattice sensAI accelerate time to market for popular use cases like object counting and presence detection.

Among the customers using the new version of Lattice sensAI is Pixcellence, a developer of image processing and computer vision solutions with advanced features like color night vision. According to Pixcellence, interest in the IoT is fueling demand for smart cameras that support AI applications like presence detection or facial recognition. The problem is that smart cameras have strict power consumption and cost requirements that make it a challenge to use

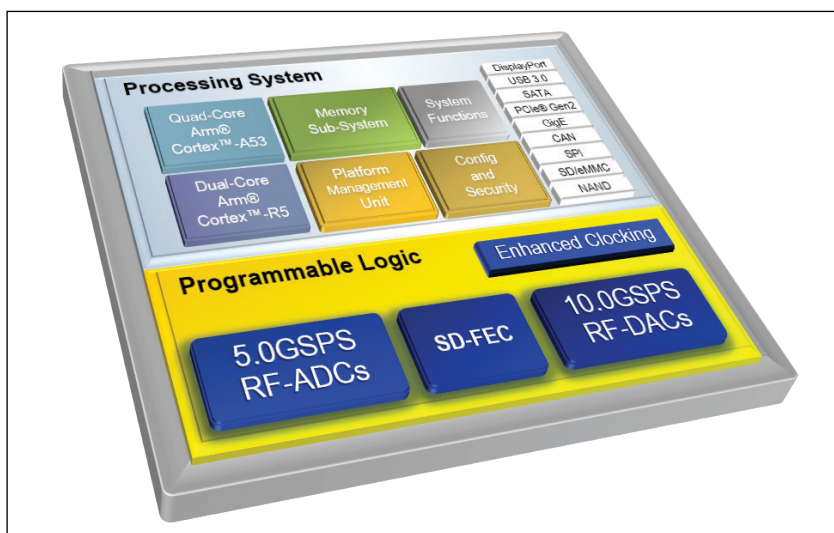


FIGURE 1

The Zynq UltraScale+ RFSoc Gen 3 device provides full sub 6 GHz direct-RF support, extended millimeter wave interface and up to a 20% power reduction in the RF data converter subsystem compared to the base portfolio.



FIGURE 2

The Agilex FPGA supports hardened BFLOAT16 and up to 40 Teraflops of DSP. Each Agilex DSP block can perform two FP16 floating-point operations (FLOPs) per clock cycle.

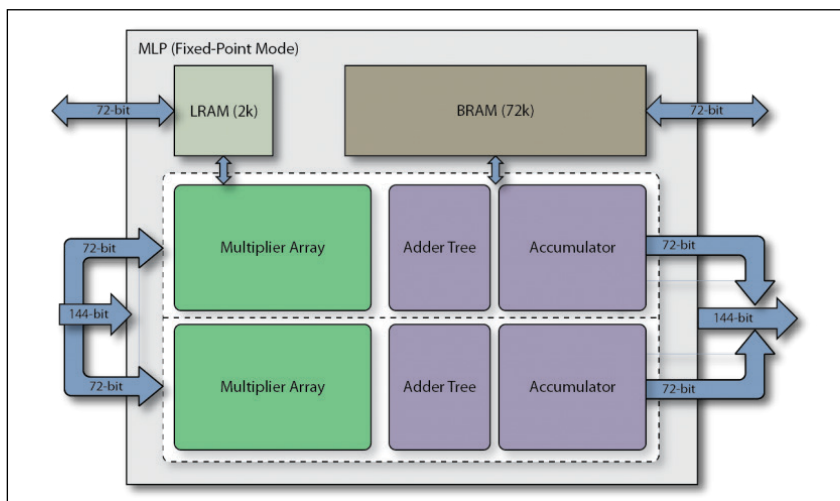


FIGURE 3

At the heart of Speedster7t FPGAs are a massively parallel array of programmable compute elements within the new MLPs that deliver high FPGA-based compute density.

off-the-shelf ASSPs. By using the Lattice sensAI solutions stack, Pixcellence said it was able to easily add low power, flexible AI inference support to its existing and new camera designs.

INFERENCE ENGINE IC

Advances in processing performance isn't only happening among the leading FPGA vendors. Vendors of Embedded FPGAs (eFPGAs) are also adding new innovations. An eFPGA is an IP block that allows an FPGA to be incorporated in an SoC, MCU or any kind of IC. Among these eFPGA companies are Flex Logix and Achronix Semiconductor. For its part, Flex Logix has always been an IP company—offering both embedded FPGAs and IP such as an inferencing IP solution it announced last Fall. According to Flex Logix, the reception of that inferencing IP was so good that it decided to develop and manufacture their own edge inferencing chip.

Announced back in April, this InferX X1 Edge Inference Co-Processor is optimized for what the edge needs—in particular, support for large models, says Flex Logix. The chip offers throughput close to data center boards that sell for thousands of dollars but does so at single digit Watts of power and at a fraction of the price. InferX X1 is programmed using TensorFlow Lite and ONNX. The device is based on Flex Logix's nnMAX architecture integrating 4 tiles for 4K MACs and 8 MB L2 SRAM. The chip connects

to a single x32 LPDDR4 DRAM. Four lanes PCIe Gen3 connect to the host processor. A GPIO link is available for hosts without PCIe. Two X1s can work together to double throughput.


MACHINE LEARNING eFPGA

The latest eFPGA from Achronix Semiconductor follows the trend toward machine learning (ML) and AI kinds of processing. In May, the company introduced its new Speedster7t family. Based on a new, highly optimized architecture, Achronix says it goes beyond traditional FPGA solutions featuring ASIC-like performance, FPGA adaptability and enhanced functionality to streamline design. Specifically designed for AI/ML and high-bandwidth workloads, the Speedster7t FPGA family features a new 2D network-on-chip (NoC), and a high-density array of new machine learning processors (MLP). Blending FPGA programmability with ASIC routing structures and compute engines, the Speedster7t family creates what Achronix dubs a new "FPGA+" class of technology.

In developing the Speedster7t family of FPGAs, Achronix's engineering team redesigned the entire FPGA architecture to balance on-chip processing, interconnect and external I/O, to maximize the throughput of data-intensive workloads such as those found in edge- and server-based AI/ML applications, networking and storage.

Speedster7t devices are designed to accept massive amounts of data from multiple high-speed sources, distribute that data to programmable on-chip algorithmic and processing units and then deliver those results with the lowest possible latency. Speedster7t devices include high-bandwidth GDDR6 interfaces, 400G Ethernet ports, and PCI Express Gen5. These are all interconnected to deliver ASIC-level bandwidth while retaining the full programmability of FPGAs.

At the heart of Speedster7t FPGAs are a massively parallel array of programmable compute elements within the new MLPs that deliver high FPGA-based compute density (**Figure 3**). The MLPs are highly configurable, compute-intensive blocks that support integer formats from 4 to 24 bits and efficient floating-point modes including direct support for TensorFlow's 16-bit format as well as the supercharged block floating-point format that doubles the compute engines per MLP.

The MLPs are tightly coupled with embedded memory blocks, eliminating the traditional delays associated with FPGA routing to ensure that data is delivered to the MLPs at the maximum performance of 750 MHz. This combination of high-density compute and high-performance data delivery results in a processor fabric that delivers the highest usable FPGA-based tera-operations per second (Tops). 

RESOURCES

Achronix Semiconductor | www.achronix.com

Flex Logix Technologies | www.flex-logix.com

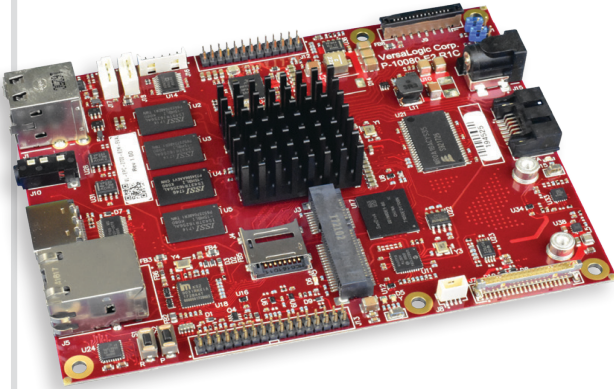
Intel | www.intel.com

Lattice Semiconductor | www.latticesemi.com

Xilinx | www.xilinx.com

Enter to Win!

Android Demo/Eval Kit



Visit circuitcellar.com/versalogic

The kit includes:

- 7" 1024x600 HDMI Touch-screen flat panel display
- Tetra Single Board Computer (SBC) with Quad-core i.MX6
- Pre-loaded Android (Oreo 8.0) on MicroSD card
- Wall power adapter
- USB Hub
- Start-up guide
- Required cables

"Set-up was a breeze,
it was up and running
in less than 10 min."

– *Electronics Technician*



VERSALOGIC
C O R P O R A T I O N

Product Focus: IoT Interface Modules

Smart Solutions

By **Jeff Child**,
Editor-in-Chief



FIGURE 1


Smart lighting is an example application that can leverage the benefits of IoT module technology

Interfacing with the Internet-of-Things (IoT) calls for highly integrated modular solutions that marry wireless connectivity and right-sized processing. The latest crop of IoT interface modules add new features such as multi-protocol support, security features and position tracking technology.

The Internet-of-Things remains full of opportunities, as organizations large and small work to develop IoT implementations across a wide range of market segments. Edge technology for the IoT requires several elements. First, it includes a combination of embedded processors and microcontrollers that provide intelligence. It also includes various wireless, cellular and other connectivity solutions to connect to the network. And, finally, it includes sensors to collect data from a multitude of IoT edge nodes.

Over the past 12 months, vendors have rolled out an interesting mix of module-based products aimed directly at IoT. These IoT products combine intelligence and connectivity, while also taking on the vital certifications needed to get IoT implementations up and running. Other emerging features in the latest IoT modules include advanced security and position tracking technologies like GNSS (Global Navigation Satellite System) receiver

technology. The product album in this month's Product Focus provides a representative look at some of these modules.

An example application that makes good use of IoT technology is smart lighting (**Figure 1**). Espressif actually offers a solution for smart lighting called ESP Mesh. ESP Mesh is a wireless communications network consisting of nodes that distribute data amongst each other. Espressif's ESP-MESH lighting solution contributes to saving a lot of energy. It is built around the ESP32-MeshKit, which is based on Espressif's ESP32 chip. In this smart lighting solution, ESP32-MeshKit-Lights can be used only whenever necessary, since they can be controlled remotely with a mobile app or a timer, a button (ESP32-MeshKit-Button), a lighting sensor or even a passive infrared (PIR) detector which is embedded in the newest development board ESP32-MeshKit-PIR. The PIR makes the lights turn on automatically, when someone enters the room, or turn off when the room becomes empty. 



LoRa Transceivers Provide Scalable Wireless Remote Monitoring

Device Solutions' Cellio LoRa long-range low-power wireless transceivers offer a solution for wide area, long range, end-to-end monitoring and control. Cellio is suited for single locations as well extensive enterprise configurations where there may be upwards of millions of sensors and controls connected to equipment across the continent.

- 915 MHz LoRa wireless technology
- Internal wireless antenna operates greater than 1 mile Line-of-Sight
- 4-5 years of operation on single 9 VDC lithium battery
- Support for 4-20 mA sensors
- Real time alert/fault detection and notifications
- Waterproof and dustproof enclosure (IP68)
- 1-wire sensor support

Device Solutions
www.device-solutions.com



RF Modules Feature Micro Form Factor

Digi's XBee3 series of next-generation RF modules and cellular modems includes a micro form factor option. Included in its micro size, Digi XBee3 also offers MicroPython programmability and dual-mode radios capable of providing RF connectivity for short range and Low Power Wide Area Network (LPWAN) applications.

- 13 mm x 19 mm form factor
- One module for all protocols including: Zigbee, 802.15.4, DigiMesh and BLE
- Eliminates the need for an external MCU
- Create smart end nodes and low-end gateways using MicroPython
- Intrinsic IoT security with Digi TrustFence

Digi
www.digi.com



Module Boasts 8 M of Pseudo SRAM

The ESP32-WROVER-B from Espressif Systems is an IoT module that comes in two versions—one with a PCB antenna and one with an IPEX connector. The module has a 4 MB external SPI flash and an additional 8 MB SPI Pseudo Static RAM (PSRAM). This marks a big improvement over the company's previous ESP32-WROVER and ESP32-WROVER-I products.

- PCB antenna and IPEX connector versions
- 4 MB external SPI flash; 8 MB SPI PSRAM
- 2.4GHz Wi-Fi, Bluetooth and BLE
- Includes dual-core ESP32-D0WD chip
- ESP32 integrates several sensors and peripherals
- Supports FreeRTOS with LwIP

Espressif
www.espressif.com

IoT Interface Modules



BLE Module Embeds Arm Cortex Processor

InnoComm BM20 module is based on Nordic Semiconductor's nRF52832 Bluetooth Low-Energy (Bluetooth LE) SoC. It features an Arm Cortex-M4 32-bit processor with FPU, 64 MHz. Target applications include smart buildings, smart homes, CE remote control, health and medical, wearables and more.

- Bluetooth 5 - Nordic nRF52832
- 32-bit Arm Cortex-M4F CPU, 64 MHz
- 512 KB internal flash, 64 KB internal RAM
- 2.4 GHz transceiver
- Supports data rates: 1 Mbps, 2 Mbps
- SPI/I2C master/slave interface

InnoComm Mobile Technology
www.innocomm.com

IoT Modules Offer Sigfox Support and BLE

Jorjin Technologies' WS2116-A0 module is powered by the dual-radio solution from ST Microelectronics based on the S2-LP ultra-low-power long-range LPWAN transceiver and the BlueNRG-2 Bluetooth low energy SoC. The chipset combination offers optimal RF performances, sleep mode, longer battery lifetime, generous flash memory up to 256 KB and 24 KB of ultra-low-leakage SRAM and a featured set of I/O peripherals.

- ST Micro BlueNRG-2 + S2-LP chipset
- Sigfox RC1 to 6
- 865 MHz to 923 MHz
- BLE up to +8 dBm output power
- Sub1G up to +24dBm output power
- Interface: UART, I2C, SPI
- Size: 22.0 mm x 22.0 mm x 2.8 mm
- -40°C to +85°C
- Certification: in progress

Jorjin Technologies
www.jorjin.com

Wi-Fi-Enabled IoT Module Has NXP LPC54018 MCU

The LPC54018 IoT module, developed by NXP Semiconductor in partnership with Embedded Artists, is a self-contained, high performance, IEEE802.11 enabled microcontroller module for development of products based on the LPC540xx MCU family. It can be used standalone or plugged into a motherboard/baseboard for rapid product development and prototyping.

- LPC54018 MCUs with advanced peripherals based on Arm Cortex-M4 core, running at 180 MHz
- High-speed device USB port
- Longsys GT1216 Wi-Fi module based on Qualcomm QCA4004
- 128 MB Macronix serial NOR flash
- Dual expansion connectors
- Reset switch
- CE and FCC Certified
- Supported by MCUXpresso Eclipse-based IDE and GNU C/C++ toolchain

NXP Semiconductor
www.nxp.com



Module Supports Bluetooth 5, Zigbee and Thread

Rigado's Bluetooth 5 (BLE), Zigbee and Thread (IEEE 802.15.4) modular solution, the BMD-340 is an advanced, highly flexible, ultra-low power multiprotocol SoM for portable, extremely low power embedded systems. With an Arm Cortex M4F CPU, integrated 2.4 GHz transceiver and an integrated antenna, the BMD-340 provides a complete RF solution allowing faster time to market with reduced development costs.

- Complete Bluetooth 5.0 and Thread (802.15.4) solution
- 64MHz 32-bit Arm Cortex-M4F CPU with 1 MB Flash & 256 KB RAM
- USB 2.0 and built in DC/DC converter for direct USB / Li-Ion power
- Transmitter certifications: FCC (USA), IC (Canada), MIC (Japan)
- Transmitter compliance: CE (Europe), RCM (Australia / New Zealand)
- Bluetooth qualified & Thread-compliant
- Sub-footprint compatible with BMD-300/301 (Nordic nRF52832)
- Dimensions: 10.2 mm x 15.0 mm x 1.9 mm

Rigado
www.rigado.com



Module Integrates Wi-Fi and Bluetooth LE 5.0

The WL865E4-P from Telit is a fully integrated dual band, dual mode, combo Wi-Fi (802.11 a/b/g/n) / Bluetooth Low Energy (BLE) 5.0 module. The module is based on an integrated tri-core system-on-chip, with dedicated CPUs for IoT application, Wi-Fi and BLE. The IoT application processor runs on Arm Cortex-M4F at 128 MHz with 300 KB of dedicated SRAM.

- Dual band (2.4 GHz / 5 GHz) Wi-Fi module
- Bluetooth Low Energy (BLE5) qualified
- Low Power consumption
- Dedicated CPU (Cortex-M4) for IoT applications
- Wide range of peripherals
- Advanced security features with integrated crypto hardware
- Industrial grade temperature range

Telit
www.telit.com



5G-Ready Module Provides Advanced Security

The SARA-R5 from U-blox is a series of LTE-M and NB-IoT modules for low power wide area (LPWA) IoT applications. The module, built on the U-blox UBX-R5 cellular chipset and the U-blox M8 GNSS receiver chip, offers end-to-end security and long product availability, making it well suited for IoT applications with long-term device deployments.

- LTE-M and NB-IoT versions
- U-blox UBX-R5 cellular chipset
- U-blox M8 GNSS receiver chip
- 3GPP Release 14 support and 5G ready
- Dedicated GNSS antenna interface
- Draw less than 1 μ A in power mode
- EAL5+ High common criteria certification

U-blox
www.u-blox.com

Picking Up Mixed Signals

Variable Frequency Drive (Part 1)

Washing Machine Repurposed

By
Brian Miller

Modern appliances claim to be more efficient, but they're certainly not designed to last as long as older models. In this project article, Brian describes how he reused subsystems from a defunct modern washing machine to power his bandsaw. The effort provides valuable insights on how to make use of the complete 3-phase Variable Frequency Drive (VFD) borrowed from the washing machine.

Twelve years ago, we decided to replace our still-functional 25-year-old G.E. washer with a new "high efficiency" front-loading model. It claimed to save lots of hot water, and reduce electrical energy use. It lived up to that promise, but 10 years later it became extremely noisy. Googling this model uncovered a litany of videos on how the drive mechanism/bearings on front-loading washers were poorly designed. If you're interested, just Google the subject for yourself, but the short story is that it would not be cost-effective to repair this machine when the bearings wore out. We dropped the spin speed down from fast to slow, to minimize the noise symptoms somewhat (negating some of the efficiency gains provided by the high spin rate), and kept it working for another year or so, before it finally died.

I worry that we may fool ourselves into thinking that we are protecting the environment by purchasing modern appliances touted as being "green" or "high-efficiency." In my case, I doubt that I made the most environmentally friendly choice, because my older washer continued to work for another family member for three more years (28 in total). In contrast, the new one only lasted 12 years before it added almost 200 pounds to the local landfill.

Before I took the washer to the curb for pickup, I took it apart and saved a very nice 3-phase motor, a motor controller and the mechanical cycle timer. My thoughts were that I could re-purpose the motor for use on my

band saw. My band saw is a carpentry model, which operates at a high blade speed, suitable for wood. Lately, however, I've wanted to use it for cutting metal, which requires a much slower speed. I was hoping that the washer motor and its companion motor controller would serve the purpose. During the wash cycle, the motor turns very slowly. But in the spin cycle the motor really revs up—probably why the bearings don't last so long.

To run an AC motor at variable speeds with useable torque, you need a Variable Frequency Drive (VFD). Also, because homes are only supplied with a single-phase electricity supply, a VFD is required to generate the 3-phase power that this motor requires. With that in mind, I knew that the motor controller module from the washer must contain a complete 3-phase VFD. I was hoping that I could get it to do what I wanted. This turned out to be more complicated than I had hoped, but it certainly made for a very interesting project.

MOTOR CONTROLLER DETAILS

Initially, I was hoping for an easy solution. That is, I hoped that the mechanical timer in the washer was the real "brains" of the unit—that it would send the "dumb" motor control module "commands" via the timer switch closures. Having worked for G.E. Appliance Service Division at the start of my career, I was well-versed in the workings of mechanical timers used in washers/dryers of the day. Forty years ago, these mechanical timers were indeed the "brains" in the appliance.

Had I paid much attention to the way modern front-loading washers work, I would have realized that the wash cycle consists of many short bursts of low-speed tumbling—reversing every 20 seconds or so. The spin cycle involves a somewhat slow acceleration, from rest up to whatever final speed setting you have chosen on the front panel. Even though these mechanical timers are quite complex—containing about 10 cams—they could not possibly handle that many short and different “phases” of the wash cycle.

Figure 1 shows a block diagram of the motor controller module in this Kenmore 970-C45162 washing machine. Sears provided a very good wiring diagram/troubleshooting guide for the complete washer, but a detailed schematic for the motor control module itself was not provided, nor did I expect it. Therefore, I did some “reverse-engineering” to come up with this block diagram. In my early investigative stages, I wired up the timer switch assembly to the motor controller and manually advanced it through the various cycles, to see how the controller performed. This would have been impossible to do without the good wiring diagram Sears had provided with the machine.

This controller uses a custom, masked-ROM DSP device—the Analog Devices (ADI) ADMC326YR. This is basically an ASIC device for OEMs, based upon ADI’s ADSP-2171 general-purpose DSP device. ADI also sold EEPROM versions of the ADMC326YR—for

development purposes—when they first introduced the chip. But the EEPROM variant was not available from distributors when I checked.

While not shown in the block diagram, six or so switch contacts (driven by cams in the timer) are connected to GPIO pins of the ADMC326YR. As the timer advances and these switch states change, the DSP in the ADMC326YR generates the requisite 3-phase PWM signals needed to run the motor in the direction/speed required. However, for each “phase” of the wash cycle—which may last from 2 to 20 minutes—the ADMC326YR will modify the 3-phase PWM outputs frequently. For example, in the wash cycle, it will rotate slowly, in short bursts, for 20 s or so, then reverse. It will repeat this process for 10 to 20 minutes.

FROM DSP TO MCU

Long story short, there was no way to use the ADMC328YR DSP for my purposes. What I need is to be able to select certain speeds within the overall range of the motor, and have the motor maintain this indefinitely, until I select another speed or shut it down. Also, given that I would be driving a band saw, I only need one rotational direction, because the bandsaw blade cuts in only one direction. However, I made provisions for both directions. I knew I would need to replace the ADMC326YR’s firmware with a microcontroller (MCU) running my own code.

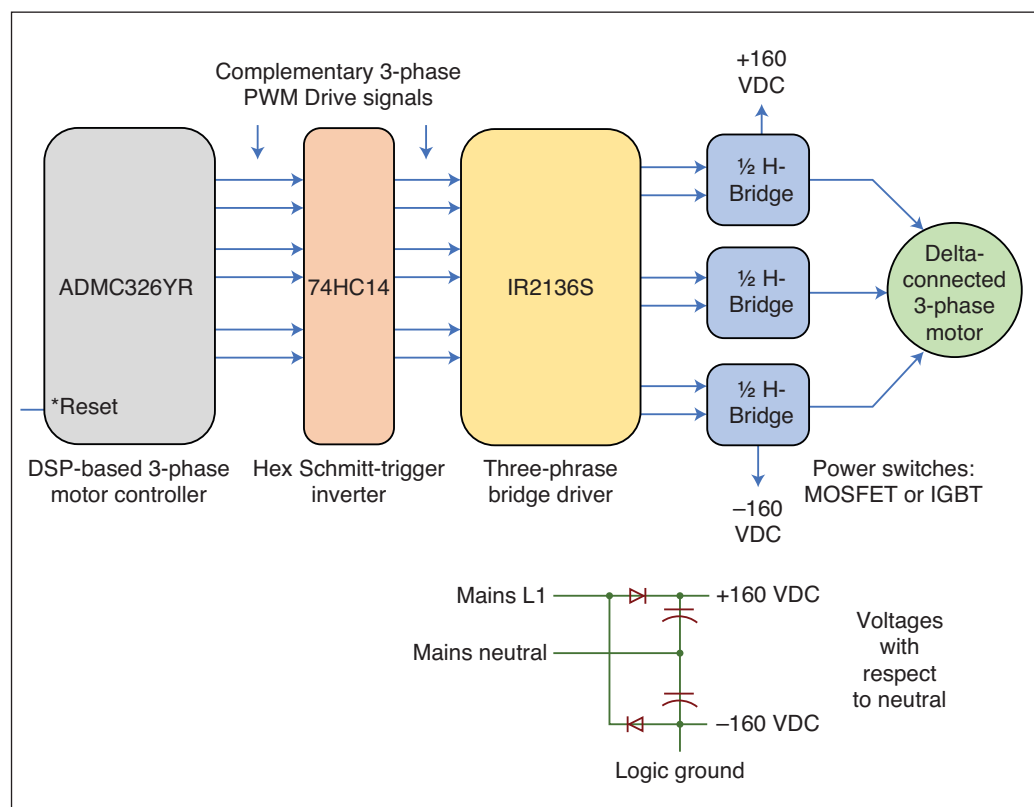


FIGURE 1

This block diagram shows the three integrated circuits that made up the washer motor’s original motor controller. Note that logic ground is connected to the -160 VDC power node (with respect to Neutral).

In the factory-designed motor controller, the ADMC326YR produces three PWM signals, each 120 degrees apart in phase. It also produces the complement of each of these three PWM signals. This is needed because each of the 3-phase PWM signals feeds a half-H-bridge power control circuit, and to do so requires two complementary signals. The ADMC326YR also handles the issue that the two complementary PWM signals (for each phase) cannot be “perfectly” complementary. If they were, there would be very short intervals during which both power control devices in the half-H-bridge would be conducting at the same time. This is due to differences in the turn-on and turn-off times of the power devices used (either MOSFETs or IGBTs), and to inherent delays in the circuitry that drives them. If both switching devices were to conduct at the same time, even very briefly, the resulting direct short across their power source would likely destroy them. The ADMC326YR inserts a “dead time” into the complementary signals to prevent this short-circuiting from occurring.

The other major active device in the motor controller is an International Rectifier (now Infineon Technologies) IR2136S 3-Phase H-bridge driver. This device converts the six TTL-level PWM drive signals into the drive signals needed to operate the six MOSFET (or IGBT) switching devices. The IR2136S contains its own “dead-time insertion” circuitry, so, strictly speaking, the ADMC326YR doesn’t really need to take care of this issue. The device also contains some current-sensing circuitry that performs over-current fault protection. The version of IR2136 used in this motor controller needs active-low PWM drive signals, whereas the ADMC326YR provides active-high PWM signals. While it’s likely the manufacturer could have used chips that had matching logic signal conventions, they chose to use a 74HC14 Hex Inverting Schmitt-trigger device to handle the logic inversion. Probably the 74HC14’s Schmitt-triggers were the more important feature, because they provide more noise immunity to the overall circuit.

The three half-H-bridge circuits used in this controller were mounted directly on the unit’s metal enclosure (for heat-sinking) and were connected to the main PCB via a wide, flexible FPC connector. Since the output switching transistors were not directly accessible without tearing the whole unit apart, I didn’t investigate whether the switching transistors used were MOSFETs or IGBTs.

In Figure 1, you can see that the HV power for the unit is obtained by a half-wave voltage doubler, providing a total of roughly 320 VDC to the half-H-bridge drive circuits. In reality, it provides ± 160 VDC with respect to the mains

(grid power) neutral line. You can see here that the Logic Common reference point (for the ADM326YR and IR2136S), is connected to the -160 VDC node. So, all the control circuitry floats at -160 V with respect to mains Neutral. Signal-conditioning circuitry in the motor controller interfaces the timer switch contacts—which operate at 120 VAC—with the ADMC326YR’s GPIO inputs, handling this difference in ground reference.

CONNECTING THE MCU

While I was designing the MCU circuitry to replace the ADMC326YR, I had to be careful how I connected my replacement MCU to the original motor controller. I definitely could not connect the motor controller to my MCU board while it was simultaneously connected to a computer via the USB port for programming/debugging. Because the computer’s USB port ground is referenced to mains Neutral, that would mean I was connecting the Neutral-referenced MCU board to the motor controller circuitry, which had its logic common referenced to -160 VDC.

This was not the only problem. Early on, I needed to determine what waveforms were being generated by the ADMC326YR. I had a rough idea, but needed more exact details, which could only be obtained by oscilloscope measurements. Nearly all oscilloscopes, including my old Tektronix analog scope and Siglent Technologies SDS 1202X digital storage scope, have their ground terminals connected to mains Neutral. So, the same ground-reference problems mentioned above existed here as well.

Luckily, I had previously built a 100 VA isolation transformer for use with my scopes. I used two identical transformers, salvaged from cordless power tool battery chargers, and wired them back-to-back. During this project, I chose to run the scope directly from the power line, and used the isolation transformer to power both the motor controller and 3-phase motor. This meant that the logic common of the controller was no longer at a potential of -160 VDC, as mentioned earlier. Instead, it was floating with respect to mains ground.

While the 100 VA capacity of the transformer was not enough to run the motor at anywhere near full-power, it was adequate for initial testing/reverse engineering purposes. As a bonus, I figured that if there were something drastically wrong with my program, the destructive power available would be limited by the 100 VA isolation transformer. I was also aware that the large filter capacitors in the power supply would negate this to some extent.

Even with the isolation transformer powering the motor controller module itself,

during development I was cautious, and programmed the MCU board in my office, using that PC, and then carried the board down to my basement shop. Whenever I hooked it up to the motor controller, my MCU board was powered by an isolated 5 V power supply.

I never seriously considered wiring my replacement MCU directly to the remaining circuitry of the original motor controller. Once my initial investigation/reverse-engineering was done, I planned to use three Silicon Labs SI8620BB-B dual isolators to couple the six PWM signals between my MCU circuit and the original controller circuit. The SI8620s easily handle the 160 VDC difference between the motor controller's logic common and Neutral and are so fast that they don't affect the timing of the PWM signals in any significant way.

You might wonder why I didn't just leave my added MCU circuitry floating at the motor controller's -160 VDC logic common potential, once the design was finalized. My additional controller circuit was going to have an LCD display, rotary encoder and some push-button switches. It was not practical to have those all referenced to something other than Neutral, for safety reasons.

Safety Note: As mentioned above, this project involves working with high voltages— ± 160 V with respect to mains ground. Also, the two large power supply filter capacitors in the original motor controller did NOT have bleeder resistors mounted across them, so high voltages persist even when the AC power is removed. I had to discharge these by hand, when doing my initial investigative work. The reader should proceed with caution if tackling a project like this.

GENERATING 3-PHASE SINE WAVES

I have used PWMs to generate DC voltages (in place of a DAC), and was also somewhat familiar with Class "D" audio amplifiers, which use PWM signals to generate high-fidelity audio. But I had never before needed to generate 3-phase sine wave signals using PWMs.

An initial Web search led me to the upper part of the diagram shown in **Figure 2**. At first glance, it would seem logical that you would generate a varying PWM signal that drove the upper gate of the half-H-bridge, using the PWM signals that are shown superimposed on the positive half of the sine wave. The lower gate of the half-H-bridge would be driven by a separate PWM signal, displaced in time by one-half of the period of the signal you were synthesizing—that is, the PWM signals superimposed on the negative half of the sine wave. Three pairs of such signals, each 120 degrees apart, would provide the requisite 3-phase AC voltages needed for the motor.

When I hooked up my oscilloscope to each of the six PWM signals coming from the ADMC326YR, it was clear that this controller did not work this way at all. Also, the assumptions from the previous paragraph do not line up at all with the way that complementary PWM signals (for motor control) are generated in any of the MCUs that I had used in the past. Instead, during the positive portion of the sinewave, you will see various duty cycles of both the "normal" and the "complementary" PWM pulses. The same thing applies for the negative portion of the sine wave. To put it another way, for the positive half cycle, the "normal" PWM output will be high 50-100% of the time, but the "complementary" PWM output will also be high for 0-50% of the time—depending on where you are within the cycle. The opposite will be true for the negative portion of the sine wave.

This can be seen in **Figure 3**, where I have superimposed both scope channels. The yellow trace is the "normal" PWM output, and the purple trace is the "complementary" PWM output. You can see where the two basically add up to 100% of the period, except for the short region that is marked by the vertical yellow cursors. This is the "dead-time" region

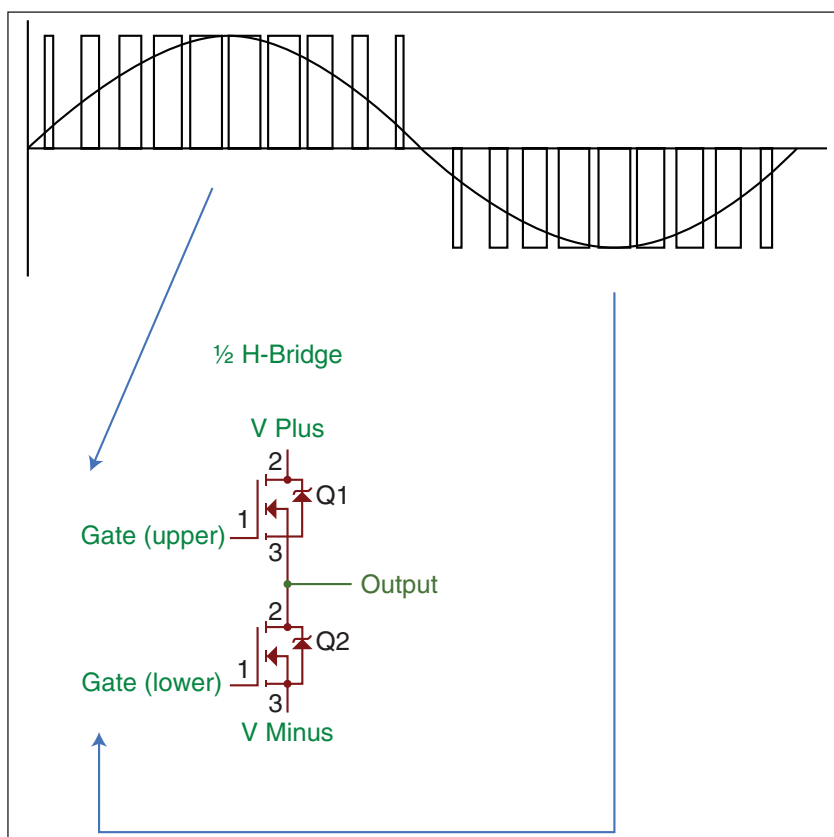


FIGURE 2

Shown here is one theoretical way to use PWM signals to generate sine waves. But it turns out it's not the way it's done in 3-phase variable frequency drives.

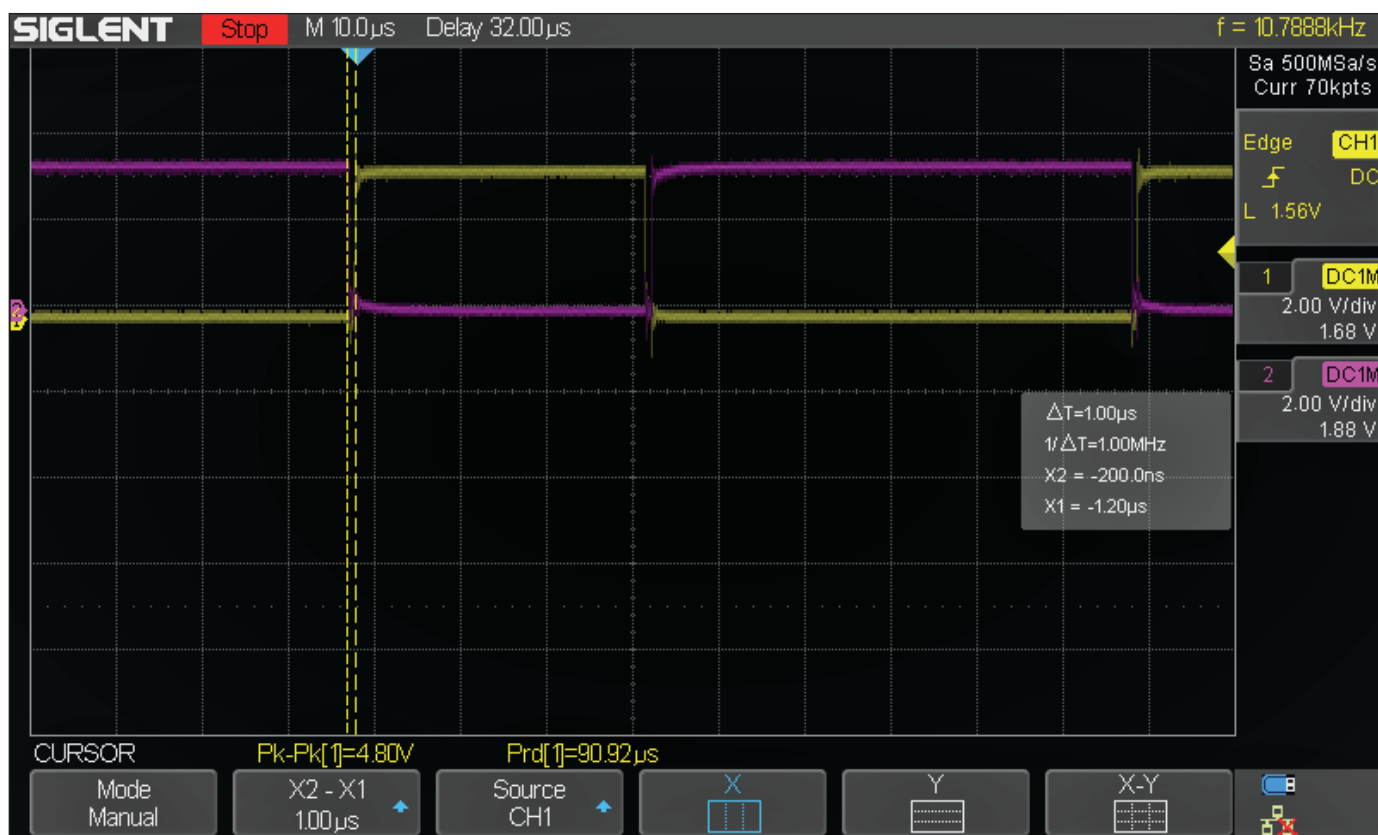


FIGURE 3

This is an oscilloscope capture of one phase worth of drive signals (normal and complementary) sent to the IR2136S. The cursors highlight the “dead time” generated by the PSoC 5LP’s PWM unit, and the PWM frequency of 10.788 kHz shows up at the upper right.

mentioned earlier. These scope traces are the actual signals that I now feed the IR2136S 3-phase H-bridge driver in the original motor controller. At the top right, you can see that the PWM frequency is 10.788 kHz. This is just one single scope capture- you can see that the “complementary” PWM output is the longer of the two. At the instant of this capture, the sine wave was somewhere in the negative half of its cycle. If you slow the scope sweep time down significantly, and take a single capture,

you can see traces like those in **Figure 4**. While you can’t make out actual PWM detail, you can discern that the purple trace is at its most dense 180 degrees out of phase with the yellow trace, as you would expect with complementary outputs.

From this investigation, I knew that I would need to use an MCU containing three PWM circuits, each of which contained complementary outputs. I wanted the PWM resolution to be at least 8-bits, preferably more. And I wanted the overall PWM period to be the same as used in the original motor controller: roughly 11 kHz.

Doing the math, at 10-bit resolution and 11 kHz period, I needed the PWM circuits to be clockable at: $11,000 \times 2^{10}$ or 11.2 MHz. This is well within the clock speeds of the PWMs contained in various MCUs that I had in mind to use.

I started this project between Christmas and New Year’s. With the holidays, I couldn’t get parts the next day from Digi-Key, as I usually do. While I had several different MCU boards on hand, I had only a few single-channel isolators, and none of the SI8620 isolators. So, my initial experiments were made without the isolators—very carefully!

For detailed article references and additional resources go to:
www.circuitcellar.com/article-materials

RESOURCES

Analog Devices | www.analog.com
 Cypress Semiconductor | www.cypress.com
 Digi-Key | www.digi-key.com
 Infineon Technologies | www.infineon.com
 NXP Semiconductor | www.nxp.com
 Tektronix | www.tektronix.com
 Siglent Technologies | www.siglent.com
 Silicon Labs | www.siliconlabs.com

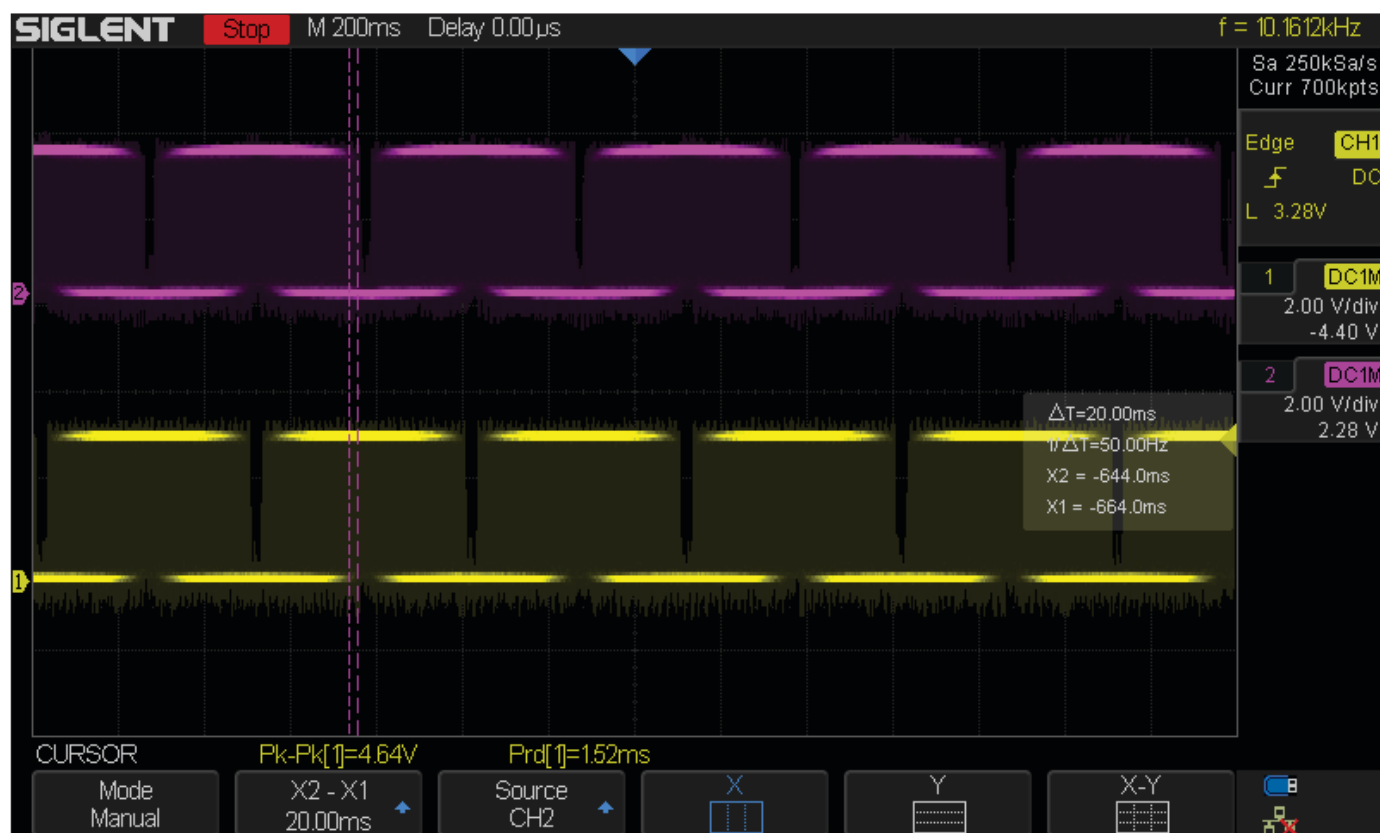


FIGURE 4

This shows multiple periods of the same two PWM signals shown in Figure 3. Although you can't discern actual PWM waveforms, you can see that the portions of the upper trace ("normal" PWM signal) that have a high duty cycle are 180 degrees out of phase with similar portions of the lower channel (complementary PWM signal).


LOGIC-LEVEL TRANSLATION

Besides isolation, the SI8620s can also provide logic-level translation. In the original motor controller, the IR2136S device and the 74HC14 inverters that feed it are running at 5 V logic levels. Most Arm MCUs, including the Teensy 3.x boards that I frequently use, operate at 3.3 V power and logic levels. The NXP Semiconductor Kinetis MCUs found on Teensy boards contain a fancy FlexTimer module that will easily do 3-phase PWM motor control—including dead-time insertion and fault control. I quickly wrote the code to try it out, and the waveforms looked fine on the scope. Unfortunately, the 3.3 V logic levels would not drive the 74HC14 on the motor controller. When the SI8620 isolators finally arrived, they would have taken care of this logic level issue, but I was too impatient to wait.

In the past, I had done some projects using Cypress Semiconductor's PSoC family of Programmable System-on-Chip devices. The newer PSoC4 and 5LP devices are Arm-based, but they can be powered by and accept logic-level signals at 5 VDC. These PSoC devices are unique in that you can basically design how you want their internal circuit blocks to be allocated and configured, both from an

analog and a digital perspective. What would normally be a rather complex design process involving Verilog is handled by Cypress' Creator 4.2 IDE application. It hides most of the complexity of these tasks with a much simpler drag and drop GUI interface.

For this project, the PSoC 5LP family seemed ideal. Cypress sells the CY8CKIT-059 development board for only \$10. It contains the CY8C5888LTI-LP097 MCU, and a full SWD programmer/debugger/USB serial port (on a tiny snap-off module). All the digital and analog functions needed for this project can be handled internally by the CY8C5888LTI-LP097 PSoC device, itself.

In part 2 of this article series (*Circuit Cellar* 350, September 2019), I'll describe in detail how I used the PSoC 5LP to handle both of the motor control PWM functions, and to provide a user interface. 

ABOUT THE AUTHOR

Brian Millier runs Computer Interface Consultants. He was an instrumentation engineer in the Department of Chemistry at Dalhousie University (Halifax, NS, Canada) for 29 years.

Embedded System Essentials

A Look at Cores with TrustZone-M Security Scrutinized

It's not so easy to keep with all the new security features on the latest and greatest embedded processors—especially while you're busy focusing on the more fundamental and unique aspects of your design. In this article, Colin helps out by examining the new processor cores using TrustZone-M, a feature that helps you secure even low-cost and low-power system designs.



TRUST.ZONE

By
Colin O'Flynn

You might feel like the moniker of “embedded security” is becoming the latest bullet point added to datasheets or marketing material of every chip coming out nowadays. But that doesn't mean everything is snake oil. In fact, there are many useful devices and features that have been released recently. With that in mind, I want to devote an article to talk about some of these, because they could make your life easier if you are looking for the

“best” secure devices. I also wanted to show why you can't just trust the press release (or datasheet!) for every feature.

In this article, I am going to discuss a new series of devices that feature a hardware separation of secure and non-secure code. The idea of separating security domains is a fundamental one, and it's a critical part of the idea of “defense in depth.” Defense in depth means not having only a single layer of defense—you never want someone who found

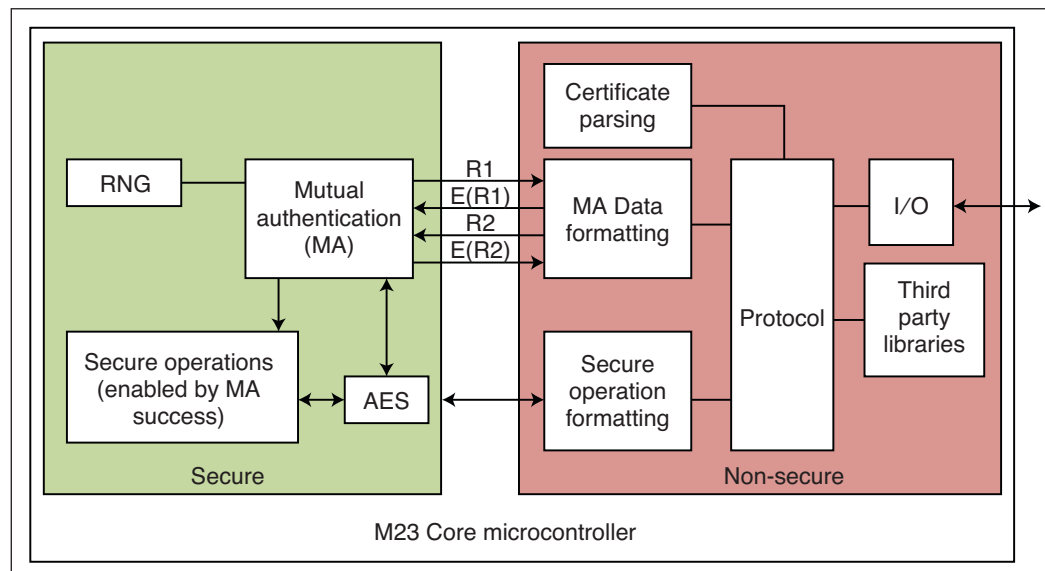


FIGURE 1

The M23/M33 core helps partition your design into secure and non-secure spaces, where limited more closely validated code goes into the secure space.

a single flaw in your system to suddenly be able to take complete control of it. The basic idea is shown in **Figure 1**. Note there is only very basic code present in the “secure” zone, whereas the “non-secure” section includes complex code such as parsing of data structures and third-party libraries which we are using.

Having a hardware barrier means that we can have a section of “highly sensitive and trusted” code. That section handles the crown jewels, such as cryptographic keys and allows debug access. This trusted code is as simple as possible—we want to ensure there is no vulnerabilities such as a bug in the command parser allows reading past the end of a data structure. By keeping complex code—which is fundamentally more difficult to secure—in an “untrusted” zone we reduce the likelihood that a security flaw in the non-secure code results in a complete compromise. Even someone with total control of the “non-secure” side has no ability to read memory in the “secure” side. With that general idea in place, let’s look at how this is implemented and used in real devices.

ON CORTEX M23/M33

This feature is present in a new core from Arm, the M23/M33 core. This includes something called TrustZone-M, which allows the separation of code into two segments like I demonstrated in Figure 1. Note that TrustZone-M isn’t a separate core—rather it enables the processor core to switch between a trusted and untrusted mode. Memory segments and peripherals can be “marked” as secure or non-secure, and such access is enforced at a low level by the core (in theory).

The M23/M33 core is unique since it represents a low- to medium-end type microcontroller (MCU) solution with those features. The M23 core is similar to a Cortex-M0+ in many respects, and M33 similar to Cortex-M4. The M23/M33 is interesting because it’s designed to avoid the “trade-off game.” In other words, just because you’re targeting ultra-low power and low cost, it

doesn’t mean you need to be stuck with poor security.

So how do you use it? For the most part, the application is developed as two separate projects with current compilers. This means an entirely separate codebase generates a separate binary that is mapped to the “secure” memory area. That binary is combined with a “non-secure” binary that can only access the non-secure code and memory spaces. Beyond just memory space, individual peripherals are also mapped into secure or non-secure code-spaces. You need to be a little careful here. Some peripherals may have non-obvious mapping between the security domains. An ADC from the non-secure space for example could measure power related to computations performed in the secure space. I demonstrated such an attack in a paper entitled “Cross-Domain Power Analysis Attacks” released last month. See the *Circuit Cellar* article materials webpage for a link.

To call between the secure and non-secure areas, a security veneer is used. This veneer is comprised of special functions that define the exact features allowed to be called by non-secure code. These would be, for example, the arrows in Figure 1 that cross between the secure and non-secure codebases.

The example in **Listing 1** shows what such a veneer looks like. Compared to some other hardware security domain solutions, this security veneer is straightforward to use because you can define arbitrary functions and use them like normal. You don’t need to worry about stuffing data through a buffer or something similar. The M23/M33 cores are a clear step in the direction of moving away from secure code feeling like a complicated mess. And they are a step towards deploying secure code being closer to regular development.

Note there are other features of M23/M33 cores beyond just the hardware barrier. For example, they include execute only memory (XOM) memory. You cannot perform a “load” operation from this memory space. If you want to prevent someone from reading out

```
/*
 * Non-Secure callable function to perform arbitrary encryption
 operations as an example
 */
void __attribute__((cmse_nonsecure_entry)) nsc_func_enc(const
uint8_t *keys, uint32_t key_len, const uint8_t *src, uint8_t *dst)
{
    //ida_u_aes_enc is a function only callable from within secure space
    return ida_u_aes_enc(keys, key_len, src, dst);
}
```

LISTING 1

Shown here is an example of the security veneer that makes it simple to move arbitrary functions into the secure code-space.

TABLE 1

A simple attempt of limited side-channel leakage, the same number of “1”s are stored in each row by always storing the bit and its complement in the silent access scheme.

W31	W30	W29	...	W3	W2	W1	W0
CompData	Data	CompData	Data	CompData	Data	CompData	Data
CompData	Data	CompData	Data	CompData	Data	CompData	Data
CompData	Data	CompData	Data	CompData	Data	CompData	Data

your code, XOM means that the only source able to read the memory is a path leading to the instruction decode logic. Using XOM requires compiler support, because, for example, the compiler cannot use look-up tables within that region.

Beyond XOM you also find support for features such as enforcement of stack pointer bounds to prevent, or at least detect, overwriting of stack frames. Overwriting a stack frame allows an attacker to change a return address, which can be used to either execute instructions from another buffer or change the return address. Incidentally, that’s part of a Return Oriented Programming attack that I discussed in previous articles, including “The Populist Side-Channel Attack: An Overview of Spectre” in *Circuit Cellar* 334, May 2018.

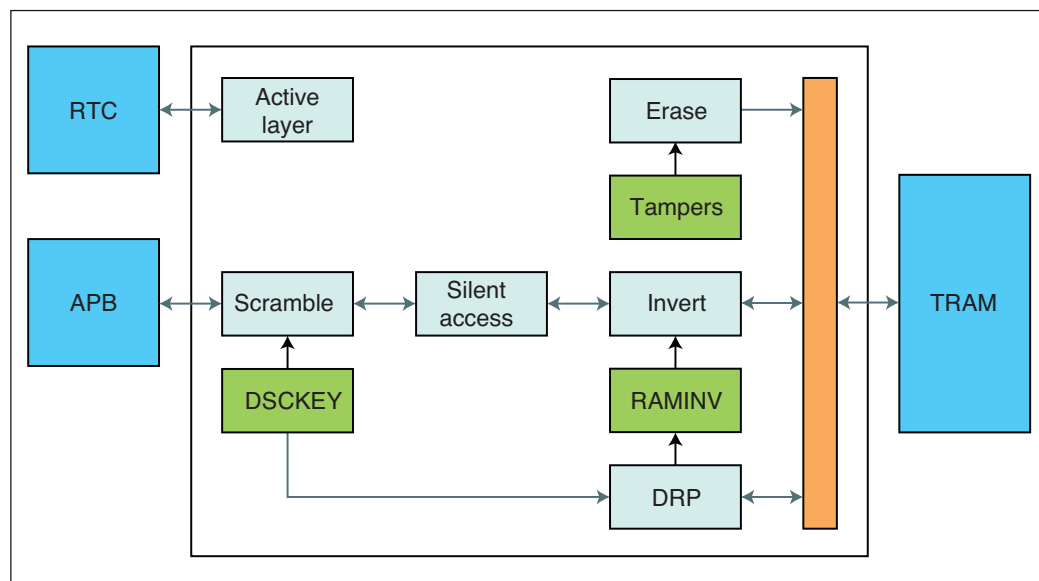
PROCESSORS WITH M23/M33

Now, with all that fluff about the M23/M33 core you might wonder: Where can you find one? The first device to market under this moniker was Microchip Technology with its SAML11 (M23 core) MCU. That was followed by the Nuvoton M2351 (also M23). The Nordic Semiconductor nRF91 was the first M33 device on the market, followed by NXP Semiconductor’s LPC55S69. Finally, STMicroelectronics (ST) is offering its STM32L5, but as of writing this article, it is not yet actually available for purchase.

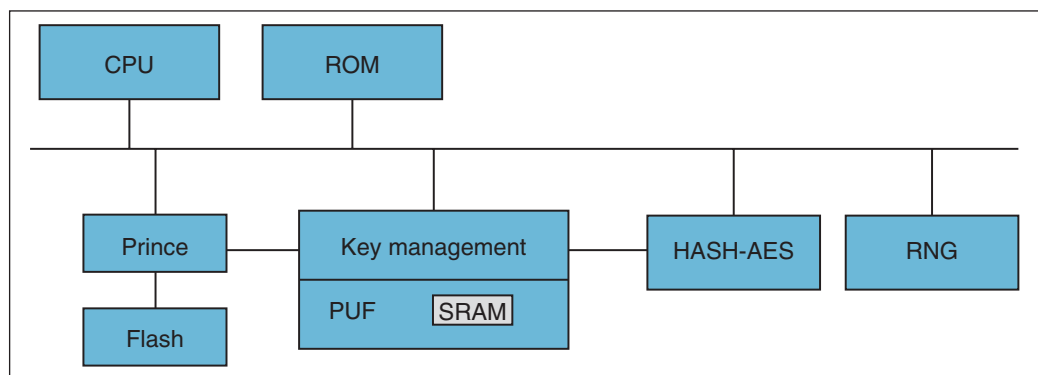
In this article, I’m going to detail two devices I’ve used: The Microchip SAML11 and the NXP LPC55S69. The Microchip (formerly Atmel) SAML11 is a M23 based device, which again should be seen as a replacement for other low-power devices such as a Cortex M0+. As you might expect it’s available in the low pin-counts and small packages that is typical for such a part. On top of the M23 core, it adds several interesting security features that might jump out at you.

One of particular interest is something called “silent access.” If you’ve read this column regularly, you’re already aware of my various side-channel power analysis demonstrations. Silent access is an attempt to reduce the effect of this—it effectively halves the usable memory for the region enabled. This enables you to split a 32-bit word into two 16-bit words, where the complement of each bit is also stored in memory as shown in **Table 1**.

The idea here is that you always have the same number of “1”s read from each word in memory. If you remember my various demonstrations, you’d remember that the number of “1”s read from memory is associated with the power consumption at the instant in time the read happens. Attackers use this to learn something about data being manipulated, which is often enough of a toehold to completely break many cryptographic algorithms.

**FIGURE 2**

Trust RAM (TRAM) has several security features, but some features such as silent access are made more difficult in practice. Side-channel leakages across the APB bus still occur.

**FIGURE 3**

The Physically Unclonable Function (PUF) serves as a master key, which can be directly routed to several encryption engines to prevent read-out of the key by malicious software.

In isolation this sounds great. But this is also implemented in something called Trust-Ram (TRAM) as shown in **Figure 2**. You'll notice that the data stored in TRAM also goes over the main bus—labeled APB for Advanced Peripheral Bus—which means that the data may be power-analysis resistant inside the TRAM device, but as soon as you are loading it into the main core, this falls apart.

TRAM does add some other interesting features. The “scramble” feature could help prevent data-remittance attacks, which are commonly used to read data “left over” in SRAM. In other words, when an SRAM is erased but its contents are not cleared effectively. Provided there are no leaks of the scramble key, you can be more confident that data stored in TRAM will be more difficult to recover from data stored in regular SRAM—even the SRAM available from the secure domain only.

Being the first M23 device to market, the SAML11 already has third party support in various compilers and libraries. While these cores are still relatively new, you may find it worthwhile to experiment with them using one of the development kits. I suspect that once they become more well known, the M23 core devices will become a mainstay of even low-cost IoT devices.

ON LPC55S69 AND UNIQUENESS

The NXP device is based on the M33 core, which is more powerful than the M23 core. The LPC55S69 extends that even further because it's a dual-core device. So, from a performance and pin count perspective, this device is a rather different part than Microchip's SAML11. But, like the SAML11, it includes some special security features beyond the M33 core.

The most prominent of these is the use of a Physically Unclonable Function (PUF). The idea of a PUF is that you have something integrated into the silicon fabric that allows generation of unique “fingerprints.” Rather than just use a pre-programmed fixed

identifier, which could easily be cloned, the identifier is based on various physical properties of the device itself along with input data. Typically, the PUF has the property that it generates a unique output dependent both on the physical property of the device and some given input. The LPC55S69 uses SRAM-based PUFs. After powering-on a SRAM array, it has effectively random data present. This pattern is used to generate unique keys, and by design some of those keys cannot even be read out by the code, but only routed directly into the cryptographic modules. This is shown in **Figure 3**, where the PUF module connects directly to various cryptographic modules.

Of course, this requires some consideration for how you will use the PUF and encryption engine. Since it's impossible (NXP claims) to read out the PUF key, you cannot generate firmware updates on a remote server already encrypted with this key. But you could use a much slower decryption method (asymmetric) to do the firmware update. Once you have a decrypted firmware it is re-written to flash with the PUF key.

One possible problem here is that, while the PUF key cannot be read out directly, side-channel power analysis could allow you to recover the encryption key being used. To provide some additional protection, the LPC55S69 has a special mode called AES Indexed Code Block (ICB) mode. This mode



ABOUT THE AUTHOR

Colin O'Flynn (colin@oflynn.com) has been building and breaking electronic devices for many years. He is an assistant professor at Dalhousie University, and also CTO of NewE Technology both based in Halifax, NS, Canada. Some of his work is posted on his website at www.colinoflynn.com.

is not one of the standard AES modes, but instead a much slower mode that uses the standard AES engine to generate a changing keystream. **Figure 4** shows a portion of how this works—each of the “E” blocks is a standard encryption mode. You can note how it generates various keys from k_0 to k_{q-1} from the original PUF key. Because different keys are used for every block, it substantially complicates side-channel power analysis attacks.

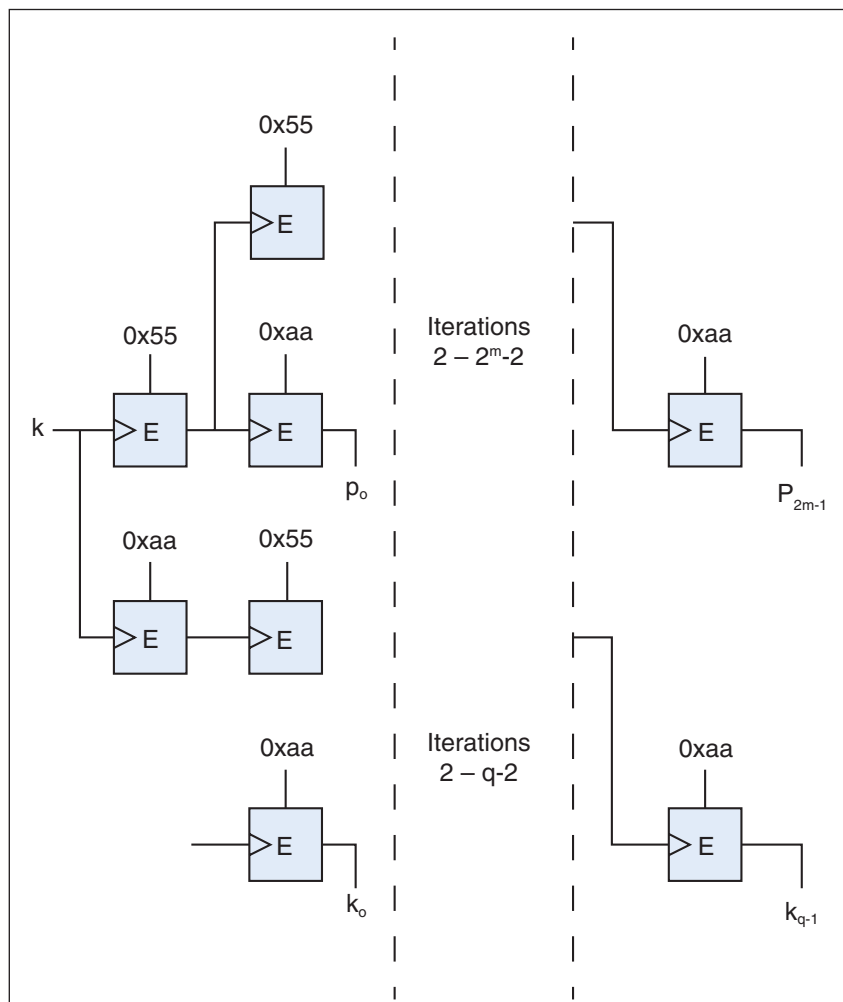


FIGURE 4

A special AES mode called Indexed Code Book (ICB) generates a keystream to make side-channel attacks more difficult, by generating keys k_0 to k_{q-1} which are then used for q encryptions.


Since the regular AES mode isn't protected against side-channel power analysis, a source of PUF key leakage is reusing the PUF key between AES-ICB mode and any other AES mode. It should be clear that taking advantage of some of these new security features requires careful usage to avoid accidentally introducing security flaws.

In addition to AES support, a hardware cipher is included specifically for encrypting/decrypting the flash memory. This is shown in Figure 3 with the PRINCE block, which is a fast stream-cipher. Again, this can be useful with some caveats. It won't protect you from an attacker reading out memory from your already running application. Presumably the encryption key is already loaded in that case, so the attacker would see the seamless decryption working as intended. But it does help prevent a case where an attacker is able to attack the bootloader or debug interface, and read out "raw" flash memory. This raw flash will be the encrypted data.

KEEP A SKEPTICAL VIEW

It's always good to remain skeptical of any new security claims. But looking at some of the new devices coming out shows that serious thought has gone into their designs. The most likely failure won't be the devices themselves, but using them in such a manner that subtly undermines the security features.

Unfortunately (for now) the manufacturers aren't keen to point out these weaknesses. But it's important to understand them so you can avoid weaknesses in your design that will be difficult to fix later. Hopefully, I've given you a useful overview of what these devices accomplish and where you might find such weaknesses, so you can go out and use them successfully.

The usage of these devices currently requires two separate applications—secure and non-secure—linked together. You might ultimately find this makes life easier to get started. If you are already using a bootloader, for example, this can be easily moved into the secure space without much effort compared to your current workflow. With a range of devices including both low-cost and high-performance, you should find something that suits your project requirements. 

Additional materials from the author are available at:
www.circuitcellar.com/article-materials

RESOURCES

NXP Semiconductors | www.nxp.com
 Microchip Technology | www.microchip.com
 Nuvoton | www.nuvoton.com
 STMicroelectronics | www.st.com



A Symposium on High Performance Chips

Memorial Auditorium, Stanford, California

August 18-20, 2019

**Register now at
www.hotchips.org**

Sunday 8/18: Tutorials

8:00 AM – 9:15 AM: Breakfast
9:15 AM – 12:45 PM: Morning Tutorials: Acceleration in the Cloud
9:15 AM – 10:15 AM: Amazon: The Nitro project – Next generation AWS Infrastructure
10:15 AM – 11:15 AM: Microsoft: Azure
11:15 AM – 11:45 AM: Break
11:45 AM – 12:45 PM: Google: TPU V3 in Google Cloud: Architecture and Infrastructure
12:45 PM – 2:00 PM: Lunch
2:00 PM – 5:00 PM: Afternoon Tutorial: RISC-V
5:00 PM – 6:00 PM: Reception

Monday 8/19: Conference Day 1

7:30 AM – 8:45 AM: Breakfast
8:45 AM – 9:00 AM: Opening Remarks
9:00 AM – 10:30 AM: General Purpose Compute
10:30 AM – 11:00 AM: Break
11:00 AM – 12:30 PM: Memory
12:30 PM – 1:45 PM: Lunch
1:45 PM – 2:45 PM: Keynote 1: Dr. Lisa Su, CEO, AMD Delivering the Future of High-Performance Computing with System, Software and Silicon Co-Optimization
2:45 PM – 4:15 PM: Methodology and ML Systems
4:15 PM – 4:45 PM: Break
4:45 PM – 6:45 PM: ML Training
6:45 PM – 7:45 PM: Reception (Wine & Snacks)

Tuesday 8/20: Conference Day 2

7:30 AM – 8:30 AM: Breakfast
8:30 AM – 10:00 AM: Embedded and Auto
10:00 AM – 10:30 AM: Break
10:30 AM – 12:30 AM: ML Inference
12:30 PM – 1:45 PM: Lunch
1:45 PM – 2:45 PM: Keynote 2: Dr. Phillip Wong, VP Corporate Research, TSMC
What Will the Next Node Bring Us?
2:45 PM – 3:45 PM: Interconnects
3:45 PM – 4:15 PM: Break
4:15 PM – 5:15 PM: Packaging and Security
5:15 PM – 6:45 PM: Graphics and AR
6:45 PM – 7:00 PM: Closing Remarks

2018 ARCHIVES NOW OPEN:

For complete presentations as well as videos go to:

www.hotchips.org/archives/2010s/hc30/

The Consummate Engineer

Energy Monitoring (Part 2)

Tracking Electric Power

By
George Novacek

In Part 1 of this article series, George began describing an MCU-based system he built to monitor his household energy. Here, he continues that discussion, this time focusing on the electrical power tracking module.

To continue with the description of my energy monitoring system, we'll now contemplate the electrical power tracking module. But I'd like to remind you once again: I challenged myself to build the system with as many components I already had in my component bins as possible to prove that engineers are a creative bunch and, as the saying goes, "can cook a meal from water."

To track the electrical energy consumption, you need to know the voltage and the current. Multiplying the two variables you obtain the immediate power $P = V \times I$. Then, by integrating the immediate power over time we'll get the total energy consumed, most often expressed in kilowatt-hours (kW-hours).

In another project—before I embarked on the design of the energy monitoring system—I tracked my household voltage. Surprisingly, after several months, I found it to be quite steady, specifically 120 ± 1.5 VAC—kudos to our power company. As a consequence, the power grid voltage in my area could be considered a constant. Therefore, only the current requires tracking.

Not having to monitor the voltage saves the additional hardware, power consumption and cost. For applications where the line voltage variation is a problem, a dual RMS-to-DC converter almost identical to the one shown

in the schematic could be used. Input current transformers would have to be replaced with voltage transformers. In addition to the extra hardware cost and power consumption one would also have to re-think the allocation of the I/O pins in order to stay with Arduino. And, obviously, the software would require modification as well.

CURRENT MEASUREMENT

The current measurement is performed by two 200 A split-core current transformers [1], one on each phase conductor as seen in **Figure 1**. This task requires that you are in the proximity of lethal electrical power. Even though installation of split core transformers is non-invasive—and you don't have to come in contact with live wires—there is no such thing as being too careful. Even for this simple and quick clamping of the transformers to the insulated power wires, I still turned off the main switch/power interrupter seen in the middle of Figure 1. And note that even turning off the breaker in the middle of Figure 1 doesn't de-energize the black wires on which the current transformers are actually being installed. Be safe! I didn't turn the power back on until all the distribution panel covers were safely back in place and affixed by their screws.

Before the transformers are clamped on the wires and the power turned on, make

sure the burden resistors are connected to the current transformer leads. I soldered them close to the transformers and covered with heat-shrinkable tubing. Without the burden resistors in place, the open circuit secondary voltage could be fairly high—the manufacturer does not specify and I didn't test it. It could damage your electronics and perhaps even cause injury.

The current transformers are rated for 200 A_{RMS} (root mean square) with transformation ratio 1:6,000. Therefore, a 200 A_{RMS} primary current flow generates a secondary current of 33.33 mA_{RMS} with $\pm 0.5\%$ accuracy and better than 0.2% linearity. This results in approximately 3.3 V_{RMS} or 4.71 V_{PEAK} across the 100 Ω burden resistor.

The current flow is sampled once every second by an interrupt-driven function. The software calculates and maintains its running average. The current average value together with a time stamp and all the other data is written into the log residing on an SD card once every hour on the hour. Once every few months I import the log into an Excel worksheet and create a graph from it. For annual assessment, logging the values once every hour results in 8,760 samples per year (8,784 in leap years), providing sufficient granularity for the data evaluation.

I will address the real-time clock (RTC) in Part 3 of this article series next month (August). The RTC is an integral part of the logger shield. It does not provide for the standard and daylight saving time changes, but it isn't difficult to write software functions to perform the appropriate changes to the monitoring log. I did it on another project, but it's an unnecessary complication for this energy monitoring system. I run the monitor on the standard time the year round.

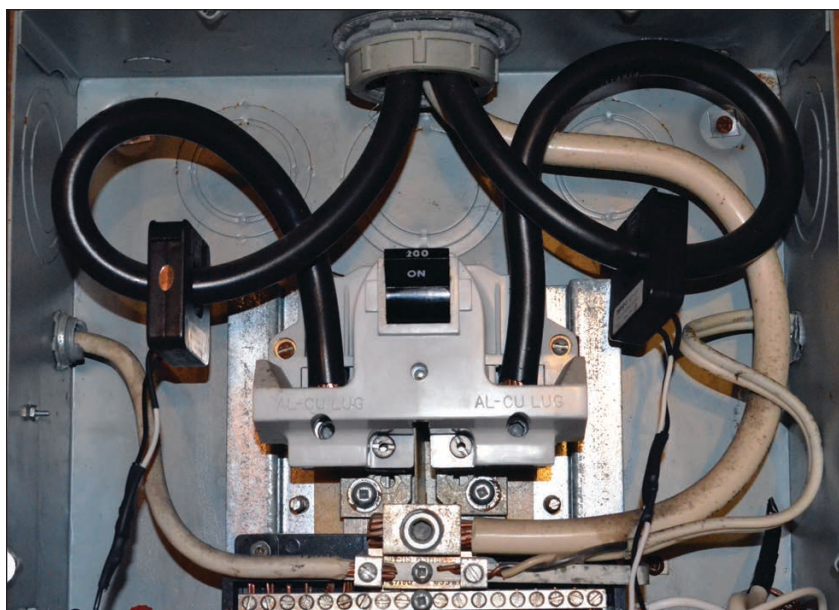


FIGURE 1

Installation of current transformers. The burden resistors are just below the transformers covered by heat-shrinkable tubing.

THE CURRENT RMS VALUE

To determine the RMS current, I chose not to use the software approach as William Wachsmann did in his project (*Circuit Cellar* 327 and 328, October and November 2017 [2]) because it is too computationally intensive and more than likely Arduino Pro-Mini couldn't handle it. As I stated earlier, minimizing the power consumption of the monitor was one essential design goal of this project and Arduino Pro-Mini draws significantly less current while providing sufficient I/O resources than most of similar devices, mainly due to the absence of the serial to USB interface.

To get the current RMS value there are dedicated ICs available for the conversion,

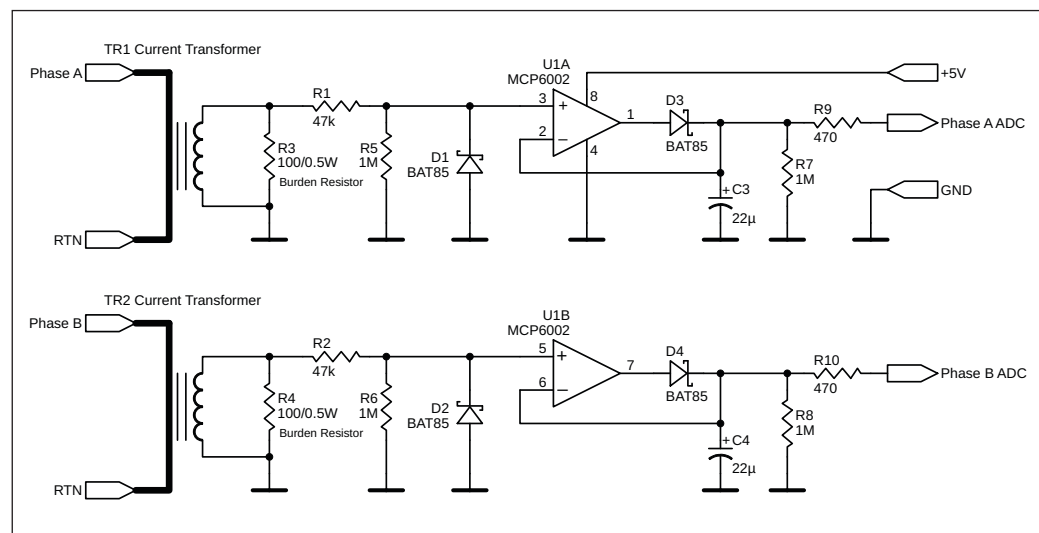
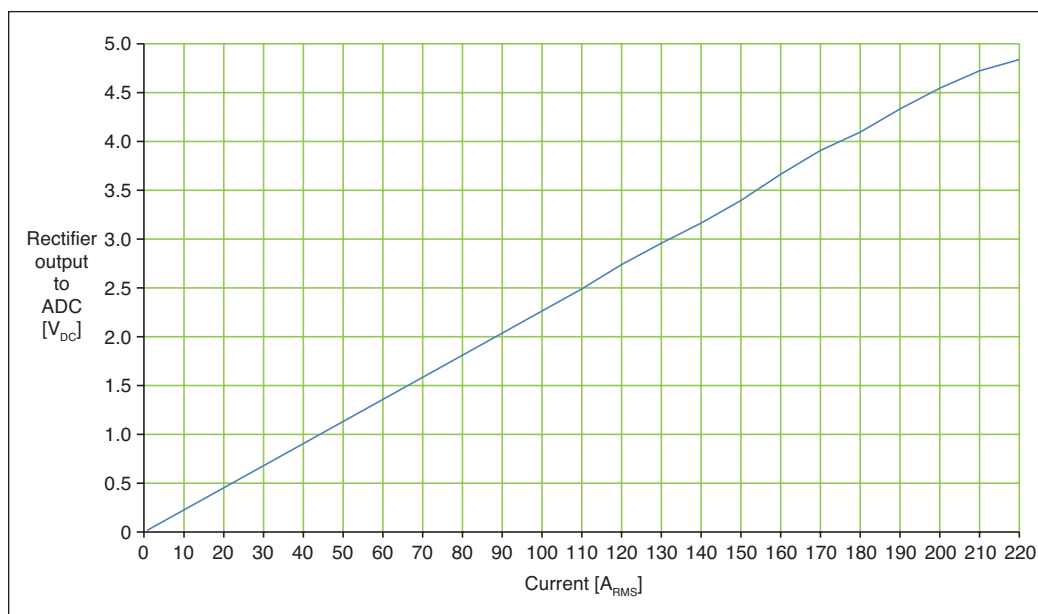


FIGURE 2

RMS to DC converter

FIGURE 3

DC convertor output to be digitized vs. the primary AC_{RMS} current



but they would be an overkill for this project. Similarly, those—and other circuits I considered using—were unsuitable for running off solar power, requiring bipolar or relatively high voltage unipolar supply. However, establishing the RMS value of a sinusoidal voltage or current for digital processing is easy. The RMS value of the transformed current generates RMS voltage across the burden resistor. By rectifying and integrating this voltage you obtain its peak DC voltage value which can be digitized.

Because the utility power is a fairly low distortion sinewave, conversions between

its different expressions—that is RMS, peak and average values—can just be performed mathematically. The RMS output voltage of the current transformer loaded by the burden resistor is calculated as:

$$V_{\text{outRMS}} = \frac{I_{\text{primaryRMS}}}{6000} \times R_{\text{Burden}}$$

Figure 2 is the schematic diagram of the circuit converting the input RMS voltage to a corresponding peak DC voltage. The peak voltage equals:

$$V_{\text{peak}} = V_{\text{outRMS}} \times \sqrt{2} = V_{\text{outRMS}} \times 1.414$$

The rectified input V_{RMS} is integrated by the storage capacitors C3 and C4 respectively and digitized by the 10-bit Arduino ADC (analog-to-digital converter). For the purpose of establishing the immediate power, the RMS magnitude is calculated from the digitized peak DC voltage by multiplying it by 0.707 (that is $1/\sqrt{2}$).

A 200 A_{RMS} primary current flowing through the current transformer will generate 3.33 V_{RMS} or 4.71 V_{PEAK} across burden resistors R3 and R4. The Microchip dual rail-to-rail op amp MCP6002 is the active element in the rectifier. Though called rail-to-rail, its usable range at 5 V supply voltage is up to about 4.5 V before some non-linearities are introduced. This can be gleaned from **Figure 3**, showing the output DC voltage versus the primary RMS current. Resistor dividers R5/R7 and R6/R8 respectively are inserted to attenuate the input AC signal to ensure the output voltage caused by 200 A primary current remains in the linear region.



ABOUT THE AUTHOR

George Novacek was a retired president of an aerospace company. He was a professional engineer with degrees in Automation and Cybernetics. George's dissertation project was a design of a portable ECG (electrocardiograph) with wireless interface. George has contributed articles to *Circuit Cellar* since 1999, penning more than 120 articles over the years.

For detailed article references and additional resources go to:
www.circuitcellar.com/article-materials

RESOURCES

Microchip Technology | www.microchip.com

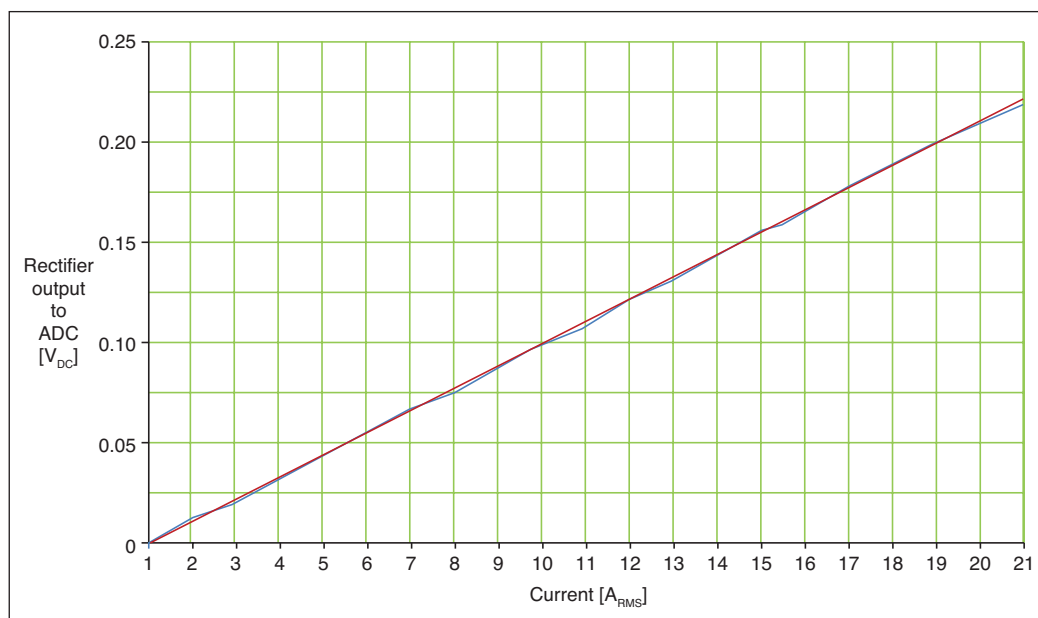


FIGURE 4

Rectification error at the input low voltage range


Diodes D1 through D4 are Schottky diodes due to their low forward voltage drop. D1 and D2 clamp the reverse polarity that would otherwise appear on the op amps' non-inverting nodes to approximately 0.25 V—well within the specification limits. D3 and D4 rectify the AC input, charging capacitors C3 and C4 to the AC peak voltage. Due to the high open loop gain of the op amps (>112 dB or approximately 400,000) and the strong negative feedback, the circuit compensates for the forward voltage drop of the Schottky diodes and rectifies the input AC signal down to 0 V input.

Figure 3 shows the relationship between the RMS current and the output DC voltage to be digitized. Because, as explained above, I reduced the input AC voltage such that the output at 200 A_{RMS} primary current does not exceed 4.5 V, there is a bit of a headroom—though not very linear—extending up to 220 A_{RMS} . As you can see from Figure 3, the peak DC output voltage is linear up to about 210 A_{RMS} current, although I would expect the circuit breaker to disconnect before such current flow magnitude is reached.

I was concerned about the rectifiers' response to the low current flow where the AC input voltage is less than the Schottky diodes' forward voltage drop. How is the linearity of the rectified output affected? The response is plotted in Figure 4 as the blue line.

When compared with the ideal red line, there seems to be just a tiny error below 2 A and around 8 A, although it could probably be due to the measurement error. That said, let's keep this error in perspective. At 10-bit digitization the theoretical system resolution is about 200 mA_{RMS} , corresponding to 24 W. The current transformers have 1% tolerance, the resistors 0.1%. At any rate, a low load

error is insignificant considering that such a low current would be a rather unusual continuous situation for any household.

Having compared my records with my utility power daily readings available from the Internet I found no adjustments or calibration to be needed. We'll conclude this project next month in Part 3. 


Verilog HDL

With the right tools

designing a microprocessor can be easy.

Okay, maybe not easy, but certainly less complicated. Monte Dalrymple has taken his years of experience designing embedded architecture and microprocessors and compiled his knowledge into one comprehensive guide to processor design in the real world.

Monte demonstrates how Verilog hardware description language (HDL) enables you to depict, simulate, and synthesize an electronic design so you can reduce your workload and increase productivity.



cc-webshop.com

From the Bench

Windless Wind Chimes (Part 2)

My MIDI Upgrade

In Part 1 of this article series, Jeff built a system to simulate breezes randomly playing the sounds of suspended wind chimes. In Part 2 the effort evolves into a less random, more orchestrated project. Using the MIDI standard, Jeff decided to craft a string of chromatically tuned chimes, similar to what an orchestra might use so the project could be used to play music.

By
Jeff Bachiochi

This project that began last month to simulate a breeze's random playing of suspended chimes has expanded—as all engineering projects seem to do. While the circuit and code presented last month can be limited to the standard five-note pentatonic chimes, in the process the code was structured to allow more than those five notes. Please read last month's article (*Circuit Cellar* 347, June 2019) to get some background on wind chimes and the music theory used to document or score how a song is written.

By the end of Part 1, you had all the tools to produce random strikes of a pentatonic chime, using solenoids instead of the wind. The aim was to give some relief during those long winter months, by bringing the sounds we relate to summer's warm tropical breezes—wind chimes—indoors. In the process of discussing the music of wind chimes and having extra I/O available on the microcontroller (MCU), I decided to create a string of chromatically tuned chimes similar to those an orchestra would use. With that in mind, the project could be used to play music as well as soothe the savage soul with random (potentially) harmonious tinkles. With 16 outputs, I can cover more than an octave of chromatic notes. A chromatic octave has seven natural notes and five sharps/flats (the

white and black keys of a piano).

When you research music, or more precisely the scoring of music, you eventually bump into MIDI. MIDI (Musical Instrument Digital Interface) is an industry-standard music technology protocol designed to create, perform, learn and share music and artistic works. MIDI communication is via a serial stream commands of what, when and how to play specific notes. The simplest MIDI stream sends commands to play and to stop playing a specific note, known as Note ON and Note OFF.

MIDI

To receive MIDI commands, we need an asynchronous serial interface running at 31.25 Kbaud (8 bits, 1 stop, no parity). You may have noticed that this is a non-standard communication rate. It was chosen to allow a 3-byte command to be sent in just 1 ms. The MIDI communication standard requires current loop circuitry to be used as shown in **Figure 1**. This isolation helps to eliminate ground loops. By rights, any device outputting MIDI should conform to this MIDI standard, but, in fact, newer technology makes this a bit outdated and USB is being written into the standard.

All MIDI commands begin with a Command Byte. The command byte has the MSB (most

significant bit) set, while all other data have the MSB cleared. This means any character from 0x80-0xFF will be a command. Please refer to the MIDI specification [1] for additional information. While the project will interpret many of the MIDI commands, I need only mention two commands that pertain to this project.

Command	Meaning
0x8?	Note OFF, where "?" is Channel (or Instrument) 0-15
0x9?	Note ON, where "?" is Channel (or Instrument) 0-15

Both commands are followed by two additional bytes—the note number and velocity—both of which are numbers 0-127. The note number indicates which note is to be played, and this is based on a piano (**Figure 2**). The velocity byte is an indication of the sound envelope or volume. Note ON with a velocity of "0" may be interpreted as a Note OFF, because it has no volume. Because our end product here is striking a chime, we really only need to pay attention to the Note ON commands because we won't be turning off a chime.

In a live performance, the musician's pressing and releasing a note will send the Note ON and Note OFF MIDI commands. If the performance is recorded, then there must be timing commands added to a MIDI

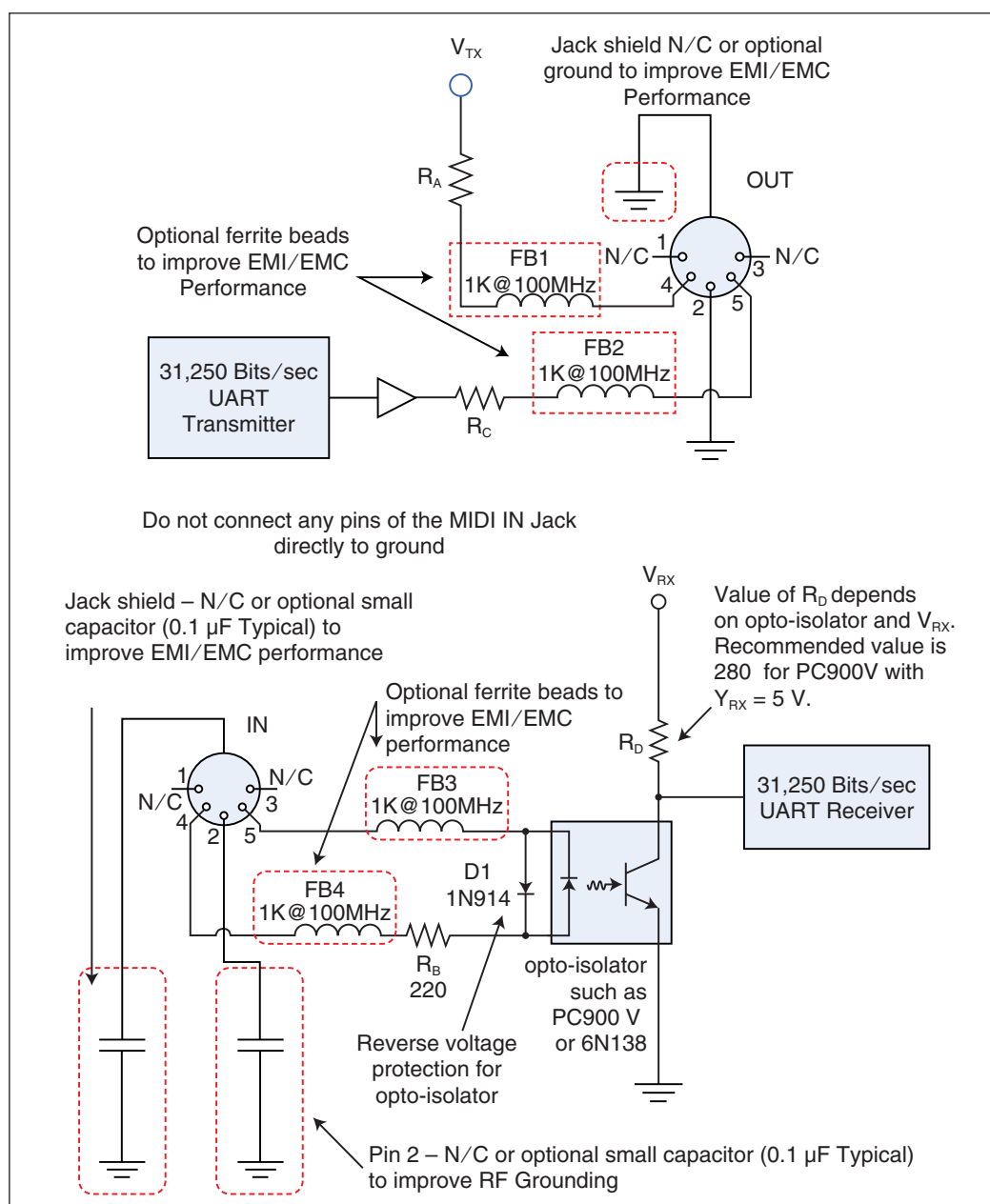
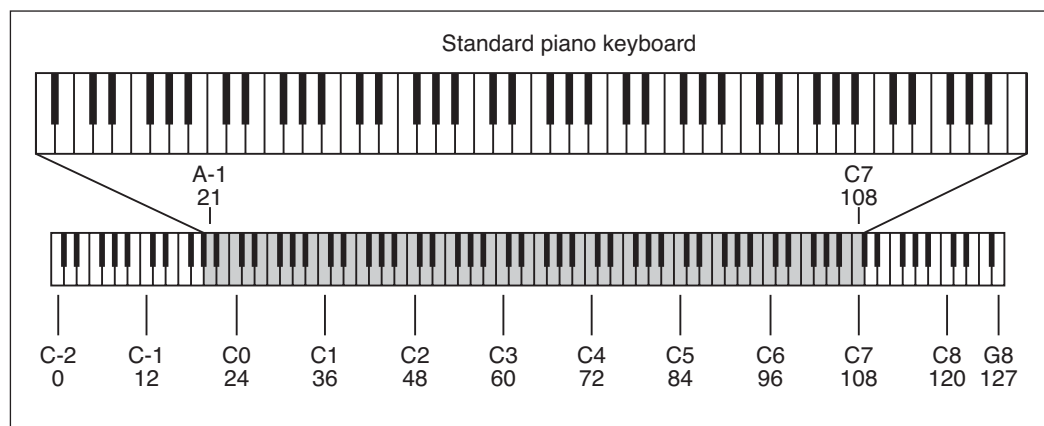


FIGURE 1

The MIDI standard uses current loop circuitry to isolate two MIDI devices from one another. This reduces the possibility of noise in one device from affecting the communication channel.

FIGURE 2

The MIDI standard uses a piano to reference note numbers. This chart shows the relationship.



file, so when the MIDI file is played back the proper Note timing can be reproduced. So, a MIDI file contains more than just ON and OFF commands. It begins with some descriptive information followed by formatted commands as a list. Each command is preceded by a delta time code. In many cases this will be a zero, meaning the following command is executed immediately. Values other than zero mean a delay must be carried out before the next command is executed. These values represent the timing added to the playing of notes to signify either a DURATION of note (before a note OFF is sent) or a REST (pause) between notes.

We've already discussed the Command rule of having the 8th-bit set and their supporting data bytes having their 8th-bit cleared. These

timing values must take on a special format, so they don't break this rule. The delta time code is made up of one or more bytes, where all bytes except the last have their 8th-bit set. This means that every byte is limited to 7 significant bits (except the last). You must squash all of the (significant) bits together to interpret the value. The value you end up with is the number of ticks per quarter note you must delay. It helps to know what a quarter note is, and you can refer to my last article or just be content in knowing that in music, this is a measure of time. Sheet music represents all notes and rests (pauses), with their own special icons representing a duration of time. That time is some division of the music's beat.

You can delve more deeply into this in last month's article. If you wish, the specifications

**FIGURE 3**

Each chime was cut to length, drilled, sanded and checked for resonant frequency. The sanding process was repeated until the chime's frequency approached the standard. Removing material raises the frequency. There's no going back.

are available at midi.org. The code for this project must understand the majority of the MIDI commands, so as not to lose sync with a MIDI file. I won't be spending time on those here. The basics for Live MIDI consist of mostly ON and OFF commands. Since this project only requires ON commands, this simplifies the execution of solenoid pulses to only those commands. This boils down to interpreting ON commands (0x9? 0x?? 0x??) by extracting the Note of interest, looking for that Note in the Note table (EEPROM), and if found, pulse the appropriate solenoid to strike the proper chime. So, now let's look into the chimes.

CHIMES

Google "DIY wind chimes," and you will find a plethora of information. My browser led me to Leland Hite's page, leehite.org. After seeing what he's written on the subject of chimes, you'll want to explore his other engineering interests. I made use of a number of charts listed there for cutting various materials into tubular chimes. I was able to find 8' of 1" aluminum stock [2] from Home Depot for less than \$15. Using the appropriate chart, I sawed up a bunch of aluminum for my chimes. See **Figure 3** for some pictures of that process. I wasn't sure of the accuracy I would need to have these chimes tuned properly. My hacksaw cuts weren't the straightest, so I added a bit to the lengths. That meant that when I used the sanding belt to square off the ends, they would end up not being too short. So, what does that mean?

As you remove material from the end of the chime, its pitch gets higher and higher. I grabbed an app for my phone that aids in tuning a stringed instrument by indicating the frequency in which the string is vibrating. I used the mobile app "gStrings" to aid in tuning the frequency of each tube (available on Google Play). It uses the phone's microphone and interprets what it hears. You need to realize that in most cases there will be overtones, or other frequencies produced that can interfere with this process. Using a soft material for rapping on the chime will produce the least interference. There are two ways in which you can tune your chime. The phone app will give you an indication of the frequency it hears, or you can set it to play the appropriate frequency, like striking a tuning fork. Listening to this while you strike the chime will allow you to judge the difference. Remember, you can only raise the pitch of a chime by shortening it. If the pitch needs to be lower, you have to cut a new chime. I was able to get pretty darn close to the actual frequency by sanding off a little at a time and rechecking as shown in **Figure 3**.

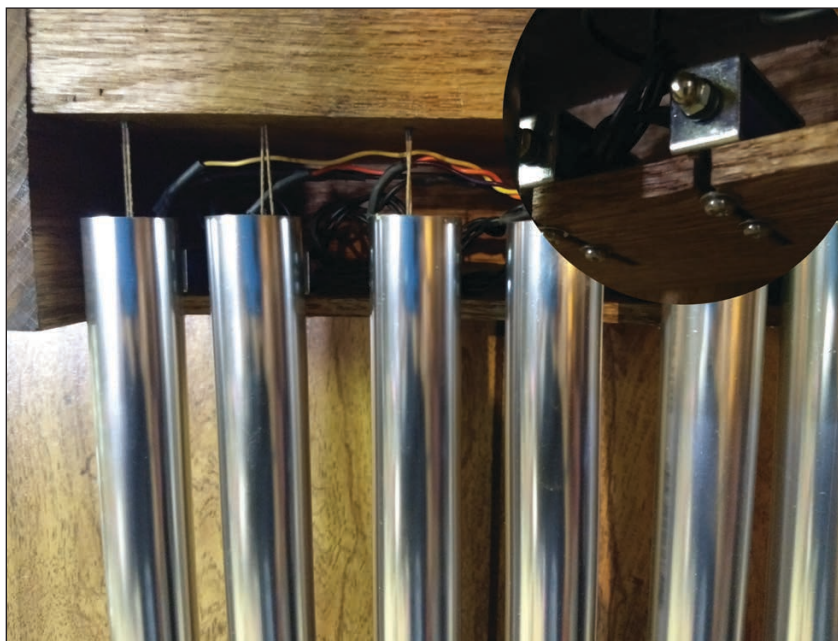


FIGURE 4

The cut and tuned aluminum tubes are suspended from an oak frame. Each tube has its own solenoid (inset) sitting behind it and ready to strike at a moment's notice.

With all the chimes cut, I needed a way to hold them and the solenoids in place. I used a few pieces of oak for the frame. Grooves at 1.5" intervals allow 1/2" between chimes (**Figure 4**). Holes in tops, centered on the solenoid grooves, allow each chime to hang in line with each solenoid's plunger. The solenoids are wired in two groups of eight, to the 9-pin connectors on the prototype PCB. It took a while to locate some #M3 hardware of an appropriate length to mount the solenoids. They're tapped for #M3 (metric screws).

If you are interested in making a traditional, round, five-note pentatonic chime, you might try using the solenoids attached to the standard suspended clapper from different directions. With this approach you would be moving the clapper like the wind does, and not striking each chime directly. However, this project's direction has taken a chromatic approach. By using chromatic



ABOUT THE AUTHOR

Jeff Bachiochi (pronounced BAH-key-AH-key) has been writing for *Circuit Cellar* since 1988. His background includes product design and manufacturing. You can reach him at:

jeff.bachiochi@imaginethatnow.com or at:
www.imaginethatnow.com.

chimes, I can reproduce notes corresponding to those played on a MIDI keyboard or other instrument. MIDI input can also direct the chimes to play prerecorded MIDI files. All this feature creep is typical of many an engineer's approach to a project. While I wouldn't do this when designing for a client, I am free here to let my projects fly free. So, while the project started off with buttons to choose modes and LEDs to provide feedback, I'm leaning toward more feedback for a cleaner "user" interface.

SSD1306 DISPLAY

The last project I used this display on was the one described in my articles "Long-Range, Low-Power Wireless Communications, Parts 1 and 2" in June and July of 2017 (*Circuit Cellar* 323 and 324). Available from Adafruit, the SSD1306 is a 128 x 64 Dot Matrix OLED display with a 4-wire I²C interface. It was simple to use with an Arduino device, thanks

to the SSD1306 library. But this time I'll need to write my own routines for the Microchip PIC MCU I'm using here. The most difficult part of using this device is getting all the registers initialized correctly for the mode in which you wish to use it. I'm using the "Page Mode" here, which allows the display to be broken up into eight, 8-row pages of 128 bits (columns) each.

When you point to a page and column with a command byte and write-a-data byte, the 8 bits of the data byte are displayed vertically in that column. A data byte of 0x00 turns OFF all the bits in that column, and a data byte of 0xFF turns all the bits ON in that column. So, with 5 bytes of the proper data, you can form any 8 x 5 graphic or alphanumeric character. You can therefore use a stored table of character data to aid in displaying messages.

Presently, I've defined messages for five of the eight pages (character lines). The top

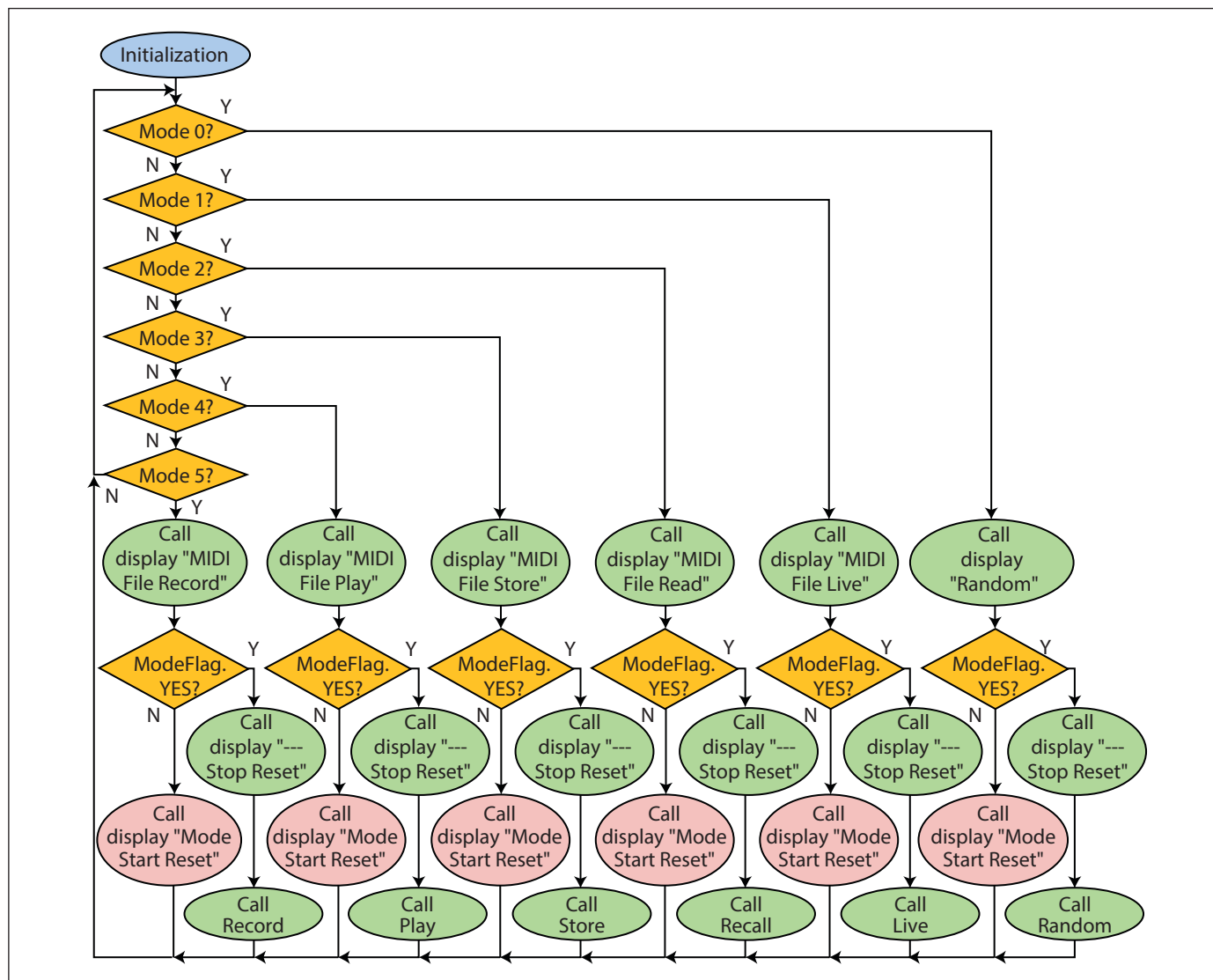


FIGURE 5

This flow chart of the "Main loop" handles the display of the menu to the user. This allows a user to select one of the six modes of operation.

two lines display typical sign-on project information. The fourth line displays the present mode. The last lines display user feedback messages that indicate which of the three original buttons are enabled and what their purpose is. One button is a reset to the circuit, and the other two are for user entry.

MENU

There are presently six modes of operation:

1. Play Random Mode
2. Play Live Mode
3. MIDI File EE Record
4. MIDI File EE Play
5. MIDI File EE Store
6. MIDI File EE Read

The first was discussed last month and is the basic “No Wind Chime.” The second allows Live Play from a MIDI device or a MIDI prerecorded file. The last four are used to capture Live MIDI, play it back, store it in EEPROM and recover it from EEPROM. These might be used to save the little ditty on board, and use it as one might use a ringtone. Button 1 allows the user to cycle through the various modes. Button 2 is used to start and stop each mode. Pretty simple, but doing this without the display would be next to impossible. Let’s see how this is all tied together by code.

CHIME CODE

Much of the MIDI code was developed to support last month’s article. I knew at the time that I didn’t want to limit the project to simply rapping chimes. The MIDI format is being used for more than just music. These days, it can be used for show, machine and phone control. This established format could use a technology update. The biggest issue would be backward compatibility. MIDI users will not accept any new standard that disregards equipment already in use. Note: To ensure compatibility, a new specification has been added MIDI-CI. This is a systematic inquiring of the equipment attached, to determine its limitations. In the case of no reply, the fallback position is MIDI 1.0. This mechanism will open up a MIDI exchange to the new MIDI 2.0 specification that is in the draft process.

While MIDI 1.0 can be complex, basic “Live” communication is pretty simple. But you need to be aware of all potential MIDI traffic, or you run the risk of losing sync with the data, at least short term. The decision to add the OLED display added a layer to the previous code. As you can see from the flow diagram in **Figure 5**, the Main loop handles this layer by determining what *mode* to

run in. The text displayed will vary slightly, depending on the mode value. Additionally, the `ModeFlag` bit “YES” determines whether the mode displayed is actually initiated. Just displaying a mode does not initiate it. While NOT initiated, Button 2’s label is “Start”. The

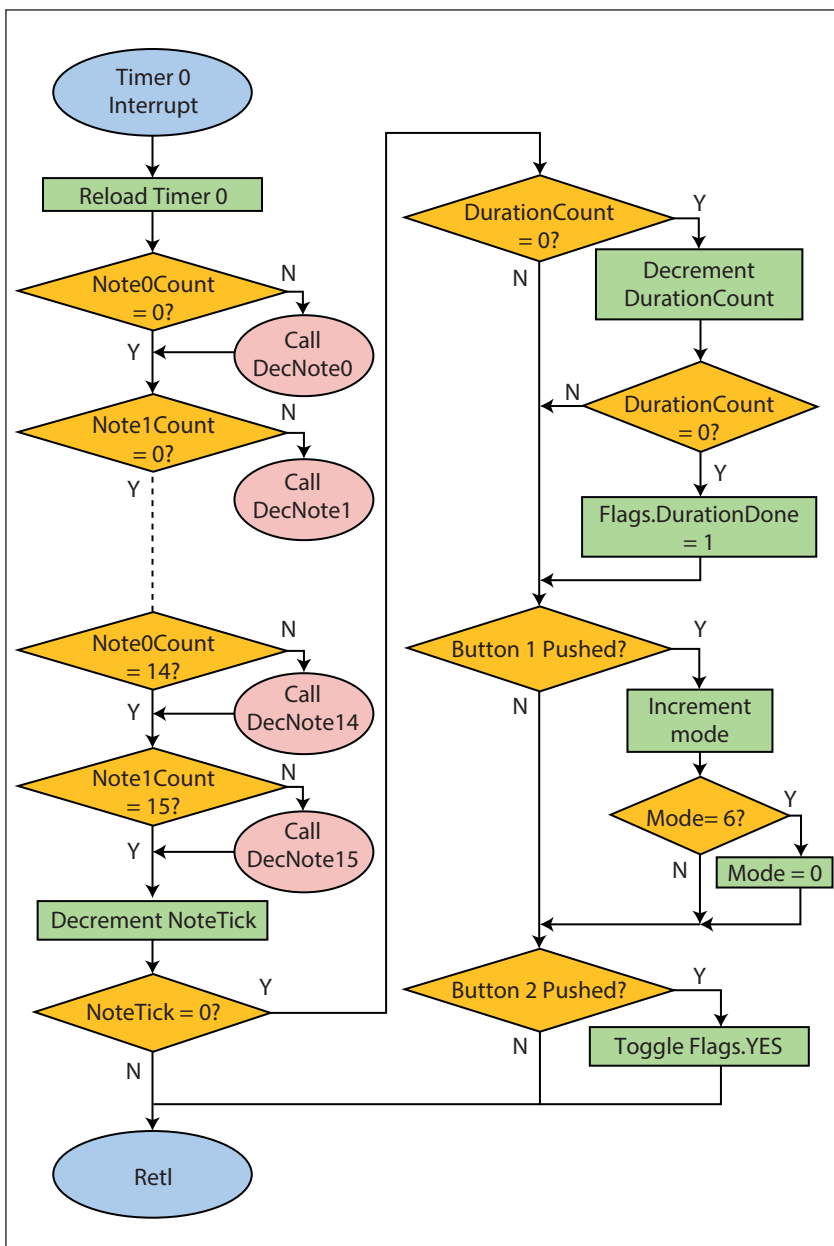


FIGURE 6

The Timer0 interrupt routine is responsible for doing all the heavy lifting, that is, handling all the timing and button presses.

Additional materials from the author are available at:

www.circuitcellar.com/article-materials

References [1] and [2] as marked in the article can be found there

RESOURCES

Adafruit | www.adafruit.com

Microchip Technology | www.microchip.com

user must initiate the mode by pressing Start. Once the mode is executing, the label on Button 2 is "Stop". Pressing Button 2 a second time will halt execution and relabel the button Start.

FIGURE 7

Playing Live MIDI requires taking in MIDI data through the serial port running at 31250 baud. Once a character is received the Process routine takes over.

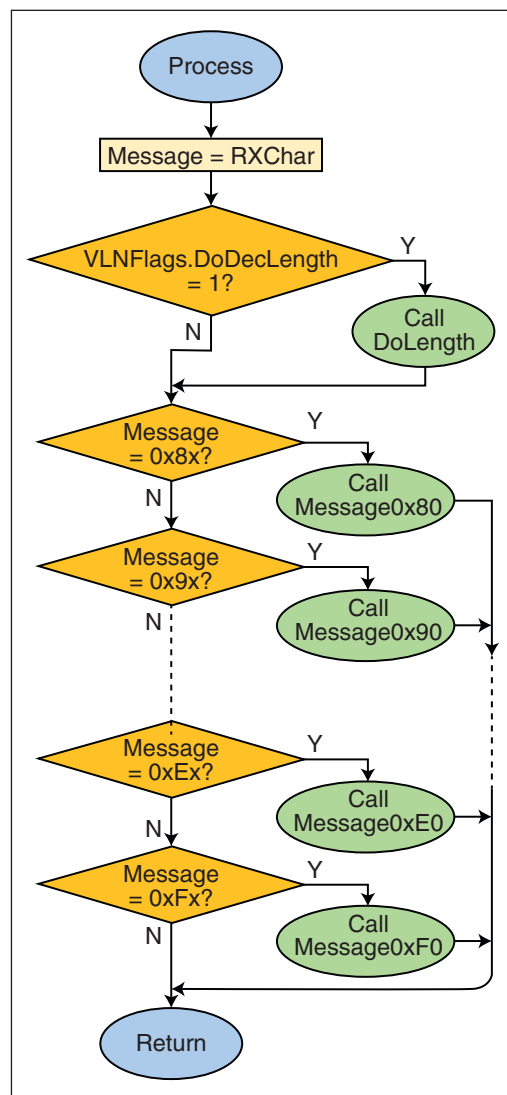
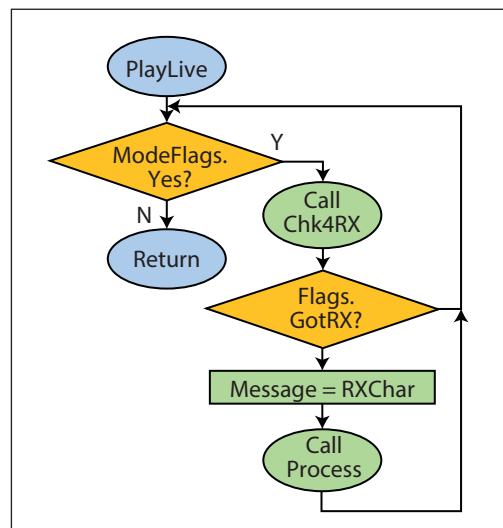


FIGURE 8

During the Process routine, we look for MIDI events or Messages. These are identified by bytes with their MSB set. There is only one Message of interest, Message 0x9?, Note ON.

Button polling is handled in the Timer0 interrupt routine. **Figure 6** shows the Timer0 interrupt, which gets triggered when the timer count rolls over. This register is loaded with a value that will roll over every millisecond. The first set of queries, Note0Count through Note15Count, handle turning off any solenoid pulse that has been activated. The activation of a solenoid also must set the associated Note?Count value to 100, to allow this part of the Timer0 interrupt to turn off the solenoid after 100 ms.

The second part of this routine only has to execute every 40 ms, so a NoteTick register—pre-loaded with 40—is decremented to keep track of this. Only when NoteTick reaches zero will the following three routines be tested. These run once every 40 ms.

40 ms is a magic number determined by the BPM (beats per minute) value. BPM is description of the tempo of a piece of music or song. It tells a musician how long to hold a note. A BPM of 120 equals 2 beats/s or 500 ms/beat. In many cases, the quarter note gets the beat, meaning a quarter note will last for 500 ms, the duration of 1 beat. Faster notes (eighth notes, sixteenth notes and so on) require faster times (250 ms, 125 ms and so forth). 500 ms /12, or about 40 ms was chosen because it is a factor of all these notes, including the special case triplets. The DurationCount can be used for timing the duration of notes or the duration of pauses (or rests) between notes. The count placed in the DurationCount register determines the time until the DurationDone flag is set to 1, a multiple of NoteTick. This is used only in the "Random Mode" where durations are chosen on the fly. For any MIDI operation, durations are part of the MIDI communications.

Finally, the buttons are polled here. The 40 ms acts a switch debounce. Button 1 increments the Mode register on each push. Button 2 toggles the YES bit in the Flags register, which starts/stops execution of the chosen mode.

PLAY LIVE

This is a bit of a misnomer. It is actually a MIDI stream that's either Live or from a prerecorded MIDI file. Note: The MIDI Record mode extracts the essentials, notes and durations and stores them in a volatile array. EE Store saves the array to EEPROM. EE Read transfers the EEPROM back to a volatile array. MIDI Play plays the volatile array. This allows the user to store a short performance and replay it. It's beyond the scope of this article to expand on these future embellishments.

There are differences in Live MIDI vs. a MIDI recording. A MIDI file has a ton of other information about the performance, whereas

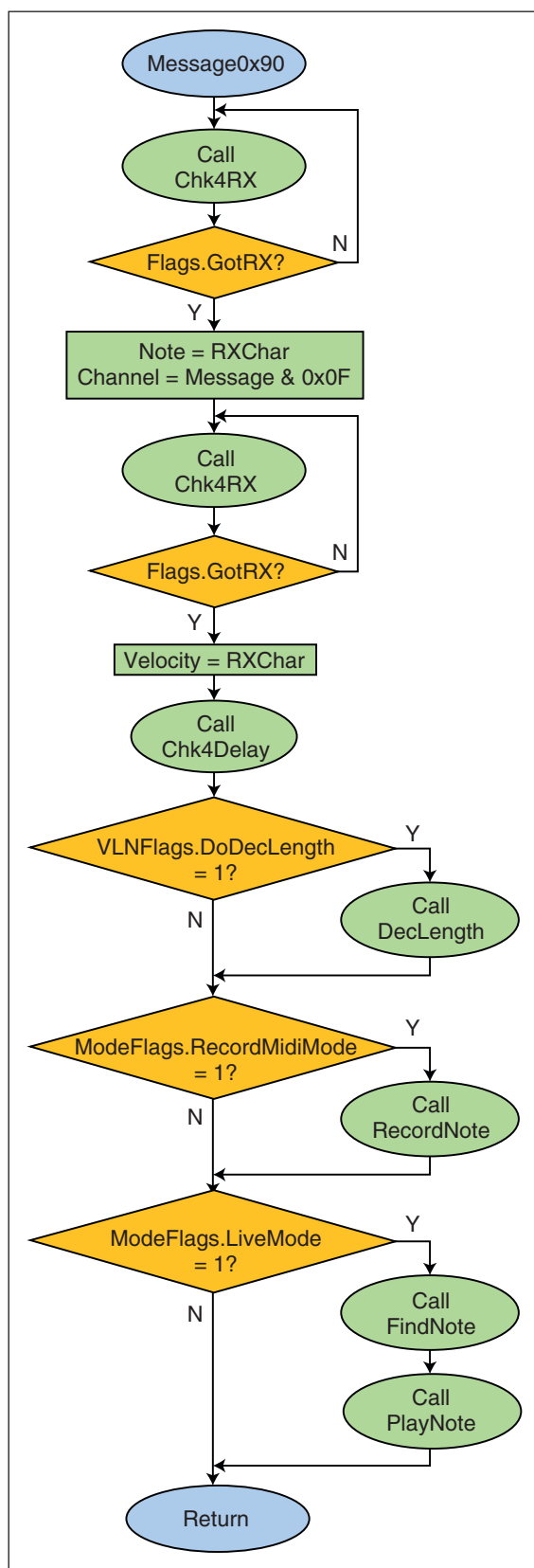


FIGURE 9

After gathering in the remaining 2 bytes of message data, we now have a “Note ON” event to process. Each of the 16 chime/solenoids corresponds to a note listed in the EEPROM Note Table. When a “Note ON” event is received, the MIDI note is searched for in the EEPROM list. If a match is found, then the corresponding note is played (solenoid is energized.)

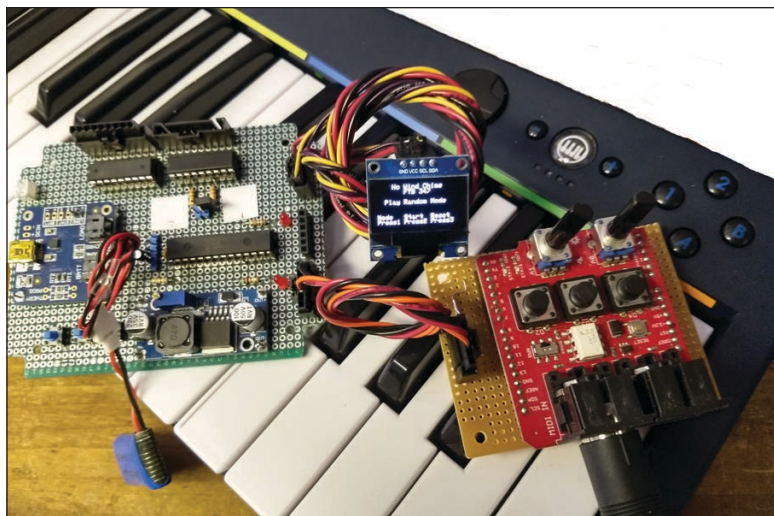


FIGURE 10

In the Live MIDI mode a MIDI output device (Keyboard) can be the driving force in activating the finished project’s chromatic chimes.

a Live MIDI stream has mostly note ON and OFF information. Note durations and rests are implied by the data within the stream. So, let’s go through what happens when the Live mode is selected and the user hit’s Start (ModeFlags.YES).

Actually, nothing happens until we receive a character, as shown in **Figure 7**. Once MIDI data begins, we can begin to interpret it. Since Live MIDI data are only event data, we know that the first byte will have its MSB set. The data will be 0x8?, 0x9?, 0xA?, 0xB?, 0xC?, 0xD?, 0xE? or 0xF?. While we are only interested in event 0x9? (Note ON), we still have to interpret all events to stay in sync with the data stream (**Figure 8**). **Figure 9** shows the flow for Event or Message 0x9?. The “?”, or LSNibble (least significant nibble) carries the channel number 0x0-0xF. The channel number or instrument allows up to 16 instruments to stream their MIDI data in the same MIDI connection. Although we could be selective here, we’ll accept any channel, and assume only one instrument is attached. Message 0x9? requires two additional bytes of data. The first is the actual Note Number (refer back to Figure 2). The second is the velocity, volume of the note. In Live MIDI we only need the Note Number, and search for it in the EEPROM Note Table—the list of notes assigned to each solenoid. If the note is in the list then it is played by pulsing the appropriate solenoid. Remember: We only need to enable the solenoid driver. The code in the Timer0 interrupt disables it after 100 ms. The execution returns (Figure 7), and we have a chance to Stop or look for more MIDI data.

PERFORMANCE

Although I’m not a musician, I do play one in this engineering column. My instrument of choice here is Rockband 3’s Keyboard (for the Wii), to stream MIDI data to this project so I can “wail some mean chimes” (**Figure 10**). I do have a pretty good imagination when it comes to performance. However, the reality is that this project is more like playing a Carillon than rockin’ out. I guess I’ll need to get a powdered wig!

Thanks for staying with me on this one. There were quite a number of different skills necessary to pull off this project—and still a lot more coding to do if I want to save a performance. Most likely, this project will get delegated to becoming a fancy doorbell or a Westminster chime. Hmm...I’ve always liked the sound of a “chiming grandfather clock!”

PRODUCT NEWS

Framework Unifies Development for PIC and SAM MCUs

Microchip Technology has announced a unified software framework with the release of MPLAB Harmony version 3.0 (v3), extending support for SAM MCUs for the first time. The development environment is progressively adding support across Microchip's entire portfolio of 32-bit PIC and SAM MCUs. This provides developers more options to meet different end application requirements. The new version also adds enhancements to streamline designs, such as royalty-free security software through a partnership with wolfSSL, as well as modular software downloads that allow designers to only download select portions of software based on the needs of an application.

MPLAB Harmony v3 provides a unified platform with flexible choices spanning architectures, performance and application focus, enabling developers to learn and maintain a single environment on their computer. MPLAB Harmony version 3.0 is available free to download on Microchip's website. The following families are fully supported in MPLAB Harmony v3: SAM E/S/V7x, SAM C21/C20 and SAM D21/D20. The following families are beta supported in MPLAB Harmony v3: SAM D/E5x, PIC32MZ EF, PIC32MZ DA and PIC32MK. All other 32-bit

SAM and PIC32 MCU families will be supported progressively by the middle of 2019. Support for new families will be added to future MPLAB Harmony versions.

Microchip Technology | www.microchip.com



4-Channel Automotive PMIC Meets Vehicle Camera Needs

Maxim Integrated Products has introduced a compact MAX20049 power management IC (PMIC) that integrates four power supplies into a tiny footprint. The device offers many options to support various output voltages, while also providing fault mitigation by flagging faults and shifts in output voltages.

Automotive camera modules tend to be size-constrained, so designers are constantly in search of a power management solution that can pack the necessary power and functionality into a small form factor. The 4-channel MAX20049 power management IC is almost 30% more compact than competitive solutions and offers the highest efficiency among other quad-power power management ICs in its class, says Maxim.

The chip offers many options to support modules that need various output voltages for different mixes of sensors and serializers, enabling designers to make changes in layout as needed or to fine-tune the IC to meet specific application requirements. The MAX20049 provides fault mitigation, a feature required by designers to help flag faults and shifts in output voltages to ensure that the cameras are working as needed.

Maxim Integrated | www.maximintegrated.com

Rugged COMe Board Sports Ryzen Embedded V1000/R1000 SoC

MEN Micro has announced the CB71C, a rugged COM Express module for rail, public transportation and industry applications. The module is 100% compatible with the COM Express Type 6 pin-out and conforms to the VITA 59 standard, which specifies robust mechanics to ensure reliable operation under the harsh environmental conditions.

The CB71C Rugged COM Express Module can be equipped with the new Ryzen Embedded R1000 SoC in addition to the Ryzen Embedded V1000 SoC. The new AMD Ryzen Embedded R1000 SoC features a Radeon Vega graphics engine with three compute units and support for up to three displays with a resolution of up to 4k without additional graphics hardware. With up to four "Zen" processor cores, when using the AMD Ryzen Embedded V1000 SoC, the CB71C is also suitable for virtualization.

The module provides passive cooling and a temperature range from -40°C to +85°C are possible with the low-power versions. The CB71C can be equipped with up to 32 GB directly soldered DDR4 main memory and a 16 GB eMMC. PCI Express 3.0, DDI (DP, eDP, HDMI), SATA 3.0, Gbit Ethernet and USB 3.0 are available as high-speed interfaces.

MEN Micro | www.menmicro.com



IDEA BOX

The Directory of PRODUCTS & SERVICES

AD FORMAT:

Advertisers must furnish digital files that meet our specifications (circuitcellar.com/mediakit).

All text and other elements MUST fit within a 2" x 3" format.

E-mail adcopy@circuitcellar.com with your file.

For current rates, deadlines, and more information contact
Hugh Heinsohn at 757-525-3677 or Hugh@circuitcellar.com.



ALL ELECTRONICS

Surplus & New Parts & Supplies
Since 1967

LEDS · CONNECTORS · RELAYS
SOLENOIDS · FANS · ENCLOSURES
MOTORS · WHEELS · MAGNETS
PC BOARDS · POWER SUPPLIES
SWITCHES · LIGHTS · BATTERIES
and many more items...

We have what you need for your next project.

Discount Prices
Fast Shipping

www.allelectronics.com



DSP ANALOG DEVELOPMENT KIT

Learn how to
EASILY
process audio
with a DSP!
Sample rates
up to 96kHz

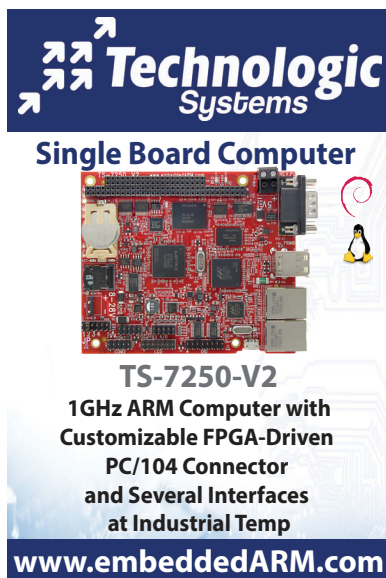
Kit includes **EVERYTHING** needed
to develop with a dsPIC®DSC:

- DSP Analog Prototyping Board
- In-Circuit Debugger/Programmer
- Power Adapters and Cables
- Clip-On Microphone & Speaker
- Exercise Book
- Single-Chip C Compiler

**ONLY
\$139!**
Use Code:
CCDSP

www.ccsinfo.com/cc719 sales@ccsinfo.com
262-522-6500 x 35

PIC® MCU and dsPIC® are registered trademarks of Microchip Technology, Inc.



Technologic Systems

Single Board Computer

TS-7250-V2
1GHz ARM Computer with
Customizable FPGA-Driven
PC/104 Connector
and Several Interfaces
at Industrial Temp

www.embeddedARM.com



Circuit Cellar 2018 Archive

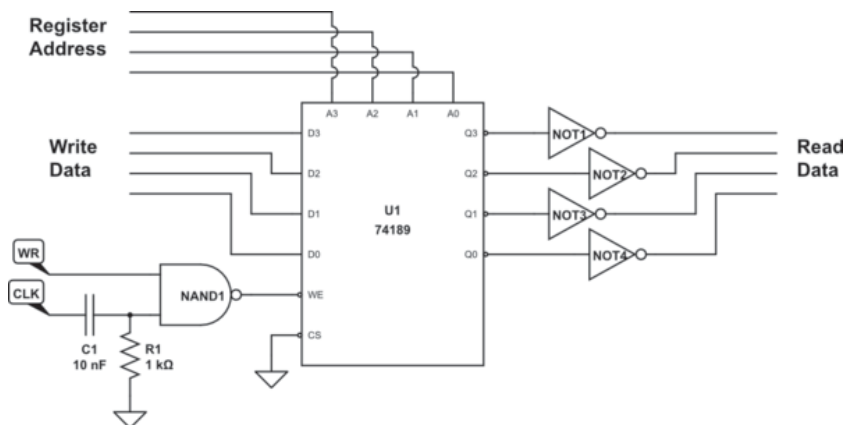
Order yours today
cc-webshop.com

Issues # 330-341 — CD #23
Copyright 2018, K&C Media Corp.

TEST YOUR EQ

Contributed by David Tweed

Problem 1— This schematic represents part of the register file for a homebrew CPU.



Problem 2— What does this tell us about the system clock frequency?

Problem 3— In image processing, what is “salt-and-pepper noise”? What causes it?

Problem 4— What is the usual method for dealing with salt-and-pepper noise? What is tricky about this kind of filter?

What is the purpose of the R-C network attached to the input of the NAND gate?

For more information:

circuitcellar.com/category/test-your-eq/

CC Vault

Unlock the power of
embedded design.

A vault of need-to-know information in the fields of embedded hardware, embedded software, and computer applications

Order yours today! cc-webshop.com

This pocket-sized vault comes fully loaded with every issue of Circuit Cellar magazine and serves as an unparalleled resource for embedded hardware and software design tips, schematics, and source code.

*CC Vault is a 16-GB USB drive.



The Future of Autonomous Cars

Sensors, Software and More Sensors



By
James Fennelly,
ACEINNA

Everyone keeps trying to predict when self-driving cars will be here—2020, 2025, 2030 and so on. In fact, Level 4 vehicles are here already [1]. Just go to Phoenix Arizona and watch a Waymo One subscriber hail a self-driving ride share vehicle. There are dozens of them operating on the roads throughout the metro Phoenix area. The challenge now is to achieve Level 4 at a cost that is not prohibitive and reaches the quality, reliability and longevity levels needed for the high-volume mass-production automotive market.

Sensors and software are the key technologies, but as of yet, the particular sensor suite solution needed for Level 4 and Level 5 implementation is still being debated. Most notably Elon Musk and Tesla lead the camp that believes LiDAR (light detection and ranging) sensors are NOT needed, while seemingly all others think LiDAR is essential.

Of the various sensor options, LiDAR sensors are the most expensive. Mr. Musk's contention is that humans do not need LiDAR to drive, so it should be possible to self-drive without LiDAR. In fact, at Tesla's recent Autonomy Investor Day, Mr. Musk indicated that Tesla's Full Self-Driving system (without LiDAR) will be feature complete by the end of 2019 and it will be robust enough to permit users to not pay attention while using it by Q2 of 2020. **Table 1** lists the majority of the sensors being used in self-driving vehicles with a short description of their primary functions, strengths and weaknesses.

THE ROLE OF GNSS

Of note, GNSS (Global Navigation Satellite System – GPS, GLONASS, BeiDou, Galileo and so on) will certainly play a part. There are a couple of enhancements or extensions to GNSS called RTK (Real Time Kinematic) and PPP (Precise Point Positioning) which greatly improves the accuracy of GNSS from a few meters down to a few centimeters. It is unclear if standard GNSS will be used to get close to real position and then the sensors on the vehicle will be used for further refinement, or if either RTK or PPP will be needed to provide the precise positioning. RTK and PPP solutions are much more costly but new chipsets are being introduced which show promise of driving the cost down to the levels needed for the mass market.

Sensors	Function	Strengths	Weakness
Radar	<ul style="list-style-type: none"> Collision Avoidance Adaptive Cruise Control Obj. Detection and Mapping 	<ul style="list-style-type: none"> Long Range Detection Velocity Determination Light Condition Independent 	<ul style="list-style-type: none"> Reflections Narrow Field of View
LiDAR	<ul style="list-style-type: none"> Collision Avoidance Adaptive Cruise Control Obj. Detection and Mapping Blind Spot Detection 	<ul style="list-style-type: none"> 360 degree View Medium Range Detection Light Condition Independent 	<ul style="list-style-type: none"> Bad Weather
Visible Light Stereo Cameras	<ul style="list-style-type: none"> Collision Avoidance Lane Keeping Traffic Signal / Sign Detection 	<ul style="list-style-type: none"> Medium Range Detection Object Discrimination Depth Perception 	<ul style="list-style-type: none"> Low Light Conditions Bad Weather Lens dirt
Infra-Red Cameras	<ul style="list-style-type: none"> Collision Avoidance Pedestrian / Animal Detection 	<ul style="list-style-type: none"> Low Light Conditions Medium Range Obj. Detection 	<ul style="list-style-type: none"> High Ambient Temp
Ultrasonic Sensor	<ul style="list-style-type: none"> Parking Assistance Traffic Following Blind Spot Detection 	<ul style="list-style-type: none"> Light condition independent Weather condition independent 	<ul style="list-style-type: none"> Short Range
GNSS / RTK / PPP	<ul style="list-style-type: none"> Navigation Absolute Position Heading, Velocity 	<ul style="list-style-type: none"> Absolute Position Accuracy 	<ul style="list-style-type: none"> Signal Drop Foliage Canopy Urban Canyon Multipath
IMU	<ul style="list-style-type: none"> Navigation / Dead Reckoning Relative Position, Heading, Velocity Vehicle Dynamics Monitoring 	<ul style="list-style-type: none"> Light conditions independent Weather condition independent 	<ul style="list-style-type: none"> Error grows exponentially with time without reference data

TABLE 1
Sensors for self-driving vehicles



FIGURE 1

Working with a suite of sensors, IMU Sensors can help ensure safe operation for autonomous cars.

All of these sensor technologies have some blind spots. The overlap of the sensor's capabilities and fusion of the data is critical to system functionality. For example, if the LiDAR is temporarily blinded by atmospheric conditions, the radar, IR (infrared) and visual light camera information can be used to maintain safe operation until conditions for the LiDAR improve. However, conditions certainly can occur that will result in insufficient data from the available sensors for safe operation. Furthermore, there is still the question as to how well the LiDAR will work when there are many LiDARs blasting light at oncoming cars also equipped with LiDAR. This is analogous to traffic coming at you with their high-beam lights on. Your view is greatly reduced.

Often overlooked but critical to supplementing these sensors are IMUs (Inertial Measurement Units). An IMU measures 3 dimensions of linear acceleration and 3 dimensions of rotational rate. From this information, attitude (pitch and roll), change in heading, velocity and position can be calculated. IMUs are used to supplement or fill in the gaps between GNSS updates and can even dead reckon position during more prolonged outages of GNSS and other sensors in the suite.

IMUs are also used to check that the vehicle movement is consistent with the driverless system input (for example, steering wheel position, brake position, accelerator position and wheel speed sensors) to detect if the car is sliding, skidding or in some other out-of-control condition not consistent with inputs, so action can be taken to bring the vehicle back under control (**Figure 1**).

IMU ADVANTAGES

The key strength of the IMU is that it works the same in all weather and geographic conditions. It is self-contained and does not suffer from degradation or outages due to weather,


mud on lenses, multipath of radar and LiDAR returns or Urban canyon effects. It is an independent data source that can be used for short term navigation and to corroborate information from other sensors. Because the sensor is independent and immune to all the other interference errors, the current trend is to view the IMU as not only a sensor to supplement and corroborate other sensors, but also to be the sensor of last resort, used to safely maneuver the vehicle out of traffic and bring the vehicle to a stop in a controlled manner when too many of the other sensors have failed or are impaired.

All vehicles in the market today equipped with ESC (electronic stability control) already have a low cost IMU (less than \$10) integrated within. However, low cost IMUs are not accurate enough to be of much benefit for inertial navigation. Precision IMUs can easily meet the performance requirements needed for inertial navigation, but today, they can cost many thousands of dollars making them not feasible for wide scale deployment in the Automotive Market. Fortunately, as is with the case of LiDAR, many companies are working on bringing the cost of IMUs with performance adequate for short term navigation into the under \$25 range. The required level of raw IMU performance is still not completely determined.

This uncertainty arises because many of the error sources for IMUs can be reduced using data from the other sensors and the target requirement is expressed not in terms of IMU performance but instead in terms of position accuracy versus time derived from the IMU data. Some commonly expressed targets are 30 cm after 30 seconds traveling at under 30 km/ hour and 20 cm after 90 seconds. There are just so many variables associated with...

- The algorithm and sensors being used to correct IMU errors.
- The algorithm calculating the position and velocity from the IMU data.
- The initial condition of the vehicle when the dead reckoning starts (velocity, turning or going straight, slowing or speeding up, going uphill or downhill and so forth).
- The path the vehicle takes after the dead reckoning starts.

...that it is difficult for the OEMs to translate this into IMU specifications.

Undoubtedly self-driving passenger cars will be equipped with many, many sensors. IMUs without question will be included. We can expect some refinement of the IMU specifications soon. In the next 9 to 18 months we can also expect to learn if Mr. Musk is successful in making a Level 4 solution available without LiDAR, and if LiDAR and GNSS (RTK or PPP) are cost reduced enough to be viable in the consumer automotive market. There is a lot of energy and money being directed in these areas so I expect significant progress to come. Fortunately, we will not have to wait very long to have answers. It is an exciting time in the auto industry. Just don't blink, because you might miss something important. 

For detailed article references and additional resources go to: www.circuitcellar.com/article-materials
Reference [1] as marked in the article can be found there

RESOURCE

ACEINNA | www.aceinna.com/inertial-systems

James received his BS EET from the University of Massachusetts. He has been working for the past 10 years with MEMS inertial sensors including component level acceleration sensors and system level products. He is responsible for defining new products at ACEINNA Inc. to meet the needs of emerging applications in the inertial sensing market.

STRONG FOUNDATION

Whether you are an
EMS, CM or OEM,
let our bare boards be the foundation
you build your reputation upon!

Technology:

Up to 50 Layers
Any Layer HDI
Sequential Lamination
Blind / Buried Vias
Laser Drilling / Routing
Heavy Copper

Materials:

Fr4
Metal Core
Isola
Rogers
Polyimide - Flex
Magtron

**We will make only what is needed,
when it's needed,
and in the amount needed.**

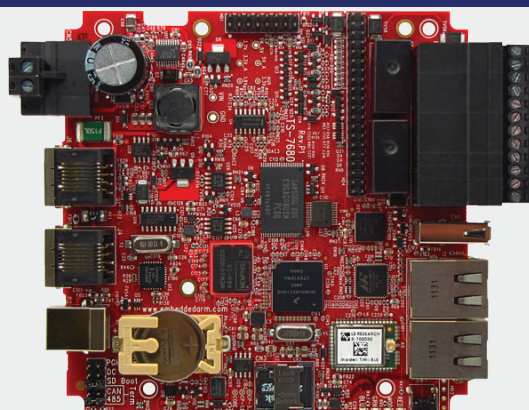
You no longer have to worry about long shelf life
or tie your capital in bare board inventory.

Accutrace[®] inc.

www.PCB4u.com sales@PCB4u.com

SAM & ITAR Registered UL E333047 ISO 9001 - 2008

FROM THE DEEP BLUE SEA TO THE WILD BLUE YONDER



TS-7680

Low Power Industrial
Single Board Computer with
WiFi and Bluetooth

\$159
Qty 100

The TS-7680 is designed to provide extreme performance for applications demanding high reliability, fast boot-up/startup, and connectivity at low cost and low power. Because there are so many features packed on to one single board computer you will see a reduction in payload weight since there is no need for additional boards, micro-controllers, or peripherals.

Rated for industrial temperature range of -40°C to +85°C the TS-7680 is deployed in fleet management, pipeline monitoring, and industrial controls and is working in some of the most demanding places on Earth.

The TS-7680 will help you perform at your very best in a variety of critical missions.



Made in USA
with Global Parts

 **Technologic**
Systems