

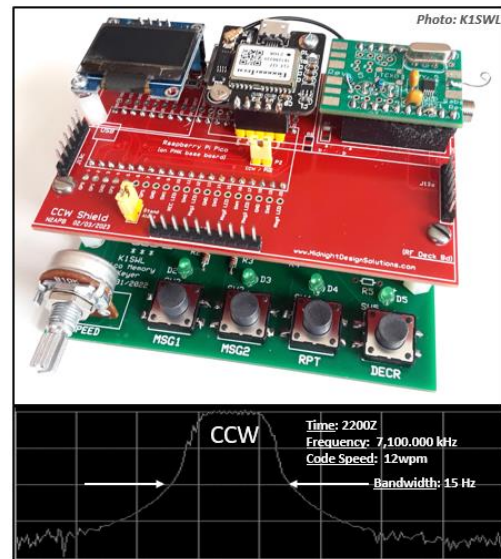
“How Low Can You Go?”

A Modern Coherent CW Transceiver for Weak Signal Communications

George Heron (N2APB) and Peter Eaton (WB9FLW)

Imagine how exciting it would be to copy a solid 599 signal when you hear nothing but the noise floor. This paper details a design of such a transceiver using an inexpensive Raspberry Pi Pico controller to achieve 20-30 dB signal path improvement over conventional CW rigs. Implementing techniques born almost 50 years ago one can turn a 5-watt QRP signal into the equivalent of more than 300 watts on the receive side with properly equipped stations.

This paper presents a brief background on the Coherent CW designs from the 1970s followed by a technical overview of our modern approach with the Pico processor including GPS-timed transmission and DSP processing of the received signal within precisely maintained windows corresponding to 12wpm code speeds ... and higher!



Introduction

More than two decades ago this paper’s authors presented on the topic of “Coherent CW” (CCW) at the FDIM 1998 seminar (Dayton Hamvention), based on original groundbreaking experimentation done over twenty years *prior to that* by Ray Petit W7GHM, et al. During that 1998 seminar we even had a haphazard collection of circuits that demonstrated the CCW principles used back in circa 1980, as documented in the very early days of the ARRL QEX periodical.

Peter and I connected again last year and while reminiscing thought how intriguing it would be if one could utilize today’s digital signal processing (DSP) technology to achieve the same performance improvements offered by the discrete logic implementations of CCW used almost half-century ago. Thus began our resurrection of understandings and the limitations of CCW. But who would possibly be interested in using Morse code at slow speeds like 12 words/minute in order to copy stations down so far “in the mud” that no other technologies today can achieve? Not even the extremely popular weak signal detection program WSJT-X has a mode for detecting or demodulating CW signals below the noise floor.

But we are getting ahead of ourselves! Most amateur radio operators do not even know what Coherent CW is; so before getting into the technical design story of the CCW transceiver itself,

let's take a slight sidestep to bring readers up to speed on the amazing principles of signal “coherency” and how it can be the absolute best friend of low power CW operators.

Coherent CW – A Quick Technical Backgrounder

There is an age-old axiom popularized in detective stories: “The more we know about something we seek the easier it is to find.” This can also apply very well to weak signal reception of Morse code. The things we need to know for schedules of course include the date and time, but equally important are **three factors enabling coherent CW reception**: the *frequency* to be used, the *timing* (wpm) of the CW signals being received, and the *phasing* of the signals or knowing precisely when to start looking for a dot or a dash. To paraphrase an aspect of the Shannon-Hartley¹ communications theorem, when we fit the narrowest bandwidth filter to the given information rate of a channel, the signal-to-noise ratio (SNR) skyrockets, which is a good thing because then only the ‘marks’ in the channel (i.e., the dits and dahs) are coming through while both the man-made band noise (QRM) and nature-made band noise (QRN) are vastly suppressed.

So in rather pseudo-technical terms, by knowing *where* to look, *when* to look and specifically *how long* to be looking – all with great stability and accuracy – we can create an extremely narrow (10 Hz!) bandpass filter for the Morse signals to yield a high SNR allowing the transmission to be easily decoded by human ear. Heretofore, if one were to try creating this extremely tight bandpass filter by using traditional techniques of analog filters or even digital/DSP filters, severe ringing and other artifacts would result, thus creating a very unpleasant and totally unusable listening experience.

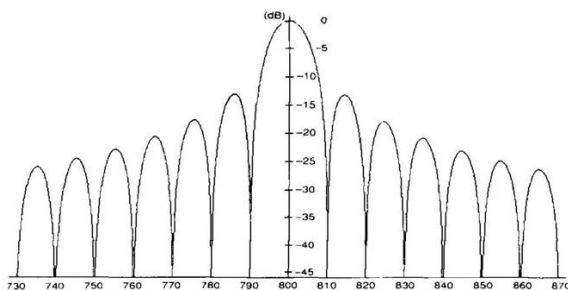


Fig. 1a

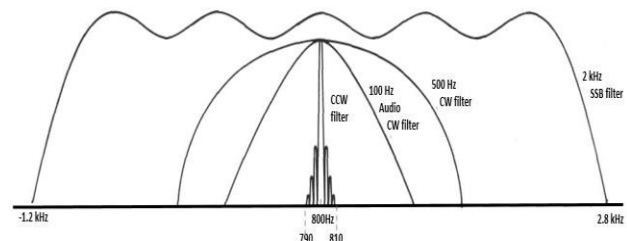


Fig 1b

Figure 1a: (LEFT) 10Hz Sampling Spectrum. The characteristic response of a 10Hz sampling filter shows notches occurring in the audio spectrum every 10 Hz.

Figure 1b: (RIGHT) CCW Bandwidth comparisons to traditional ham filters. The narrow 10 Hz CCW filter shown in the center is comparatively miniscule the traditional CW and SSB filters. One can envision the amount of QRM and QRN eliminated when using the CCW filter, and the correspondingly high signal-to-noise ratio for the precisely tuned 800 Hz CCW signal being passed.

Coming back to the genesis of coherent reception of CW, Ray Petit W7GHM published findings in the September 1975 issue of QST wherein he described how he designed integrate-and-dump circuitry to measure and determine when a mark (i.e., 800 Hz transmitted energy) was present at very specific time periods during a Morse code transmission. A very accurate 100-millisecond “timing frame” was used for the 12 wpm CW rate and WWV time-synchronization was required

¹ https://en.wikipedia.org/wiki/Shannon%E2%80%93Hartley_theorem

by the transmitting station to ensure that the frame being measured on the receive side was identical to the frame in which the mark was being sent.

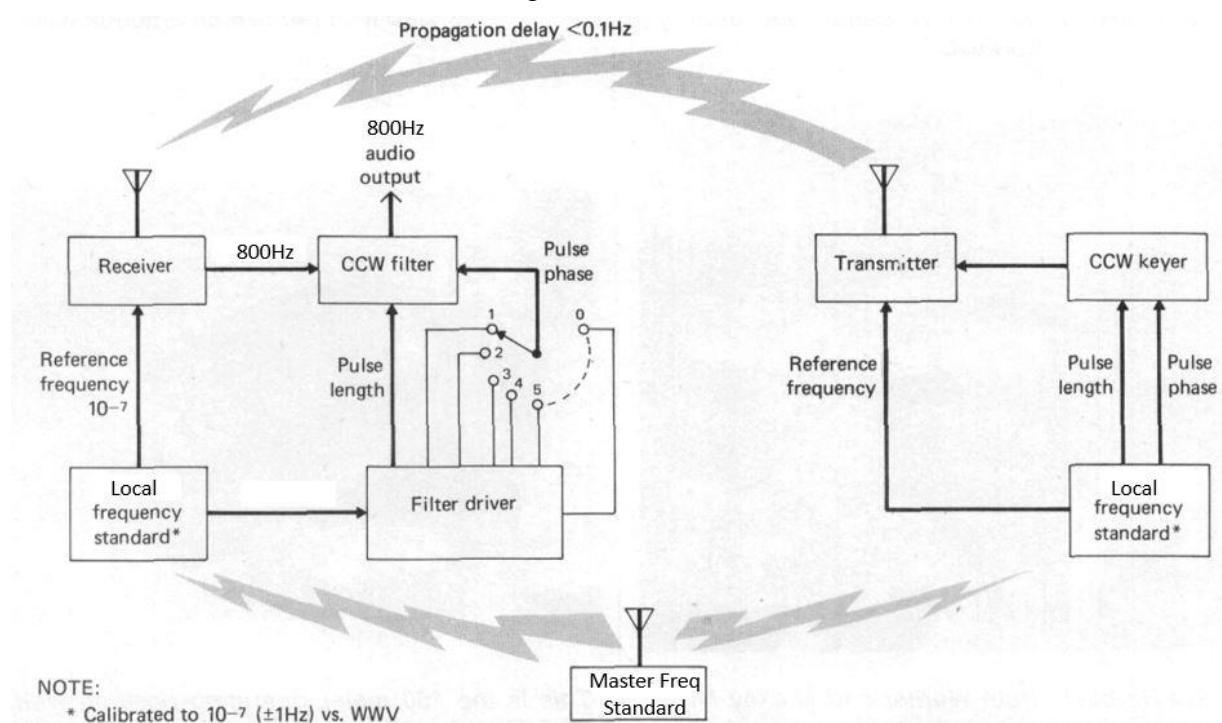


Figure 2: A CCW communications system. The three factors are illustrated that enable Coherent CW reception: known and accurate frequency, pulse length (wpm) and pulse phase (frame timing window). A system **circa 1975** used WWV for the time sync, a local oscillator as the master frequency standard, and filter driver for the phase adjust. In the **circa 2023** system we can depend on GPS for both time sync and (global) frequency standard, and digital signal processing for CCW filtering. [Graphic derived from CQ Magazine June 1977, Ade Weis “Coherent C.W. – The C.W. Of The Future Part 1”]

Of course this also meant that the transmitting station needed to ensure that the dots and dashes of were being sent precisely within those same 100-millisecond frames. Hence the sender also needed to have perfect time synchronization with WWV and very stable local oscillators of the transmitter itself.

The rest is an “exercise for the student”, as they say. The integrate-and-dump circuitry would integrate the channel receive energy during the 100-ms frame period, then dump and measure it at the end to determine the state of that frame. Based on a given threshold level of the analog sample average, the state would either be a mark or a space. When a mark was determined to be present, a tone oscillator would be sounded and the operator would hear a clear and pure note, free of any band interference or noise – solid 599 CW reception of a signal that likely couldn’t easily be heard through the receiver’s speaker!

The “fly in the ointment” back in those early days was the difficulty in achieving and maintaining stability in the oscillators used for frequency transmission and time synchronization. Any drifting would cause the CCW system to quickly fail. Others came to the rescue, with some notable success. Bill de Carle (VE2IQ) developed on these early techniques to develop some of the first BPSK (Binary Phase Shift Keying) implementations, and technology advances brought along some early embeddable DSP solutions such as what Rick Campbell (KK7B) and Johan Forrer

(KC7WW) achieved with the Motorola DSP56KEVM systems; but none really took steadfast hold within the Amateur Radio's community of CW operators.

After our initial CCW presentation at FDIM 1998 we established an online collection of all the published works we could locate concerning Coherent CW. The collection consists of over sixteen unique papers and a full collection of more than 100 issues of the CCW Newsletter, published from 1976 through 1994. To this day we continue refining the **CCW Archives** to provide a lasting picture of how this exciting technology evolved and the pioneers in our ham community who made it happen: Ray Petit W7GHM, Charles Woodson W6NEY, Bill de Carle VE2IQ, Peter Lumb G3IRM, Ade Weiss W0RSP, Stan Wilson AK0B, and Vic Black AB6SO. The address for the CCW Archives is indicated in the Bibliography and below². Some really great reading material is in those Archives!

CCW Operation – A Little Deeper Dive

So now we've established the basics: CCW utilizes the notion that signals are sent at specific times and not at somewhat arbitrary times as in CW. All CCW dots, dashes and spaces are exact multiples of a basic time unit and occur within predictable time frames, including any pauses during transmission. When received, CCW signals should be like CW signals except that they are sent very precisely. As a result, very narrow filters can be used for reception. For example, if 12 wpm is used for sending, CCW receive filter bandwidths would be on the order of 10 Hz. A 1-watt signal copied through a 10Hz filter is equivalent to a 50-watt signal heard through a 500Hz filter and a 230-watt signal copied with a 2300Hz filter. CCW indeed can offer quite an advantage for weak signal reception. This is all pretty great stuff ... so how is it done?

The heartbeat of the modern CCW transceiver is provided by a small GPS receiver module configured to deliver a 10Hz signal to a microcontroller that in turn is configured to interrupt the processor every 100 milliseconds, the Morse element timing frames for 12 wpm code speed. Within each of these frames, marks or spaces are transmitted as Morse code and then decoded as during receive.

The magic of CCW weak signal reception occurs on the receive side with precise timing of the 100ms frames and FFT processing being performed within each of those frames to determine if a mark or a space is present. Data collection is begun at the start of each frame, followed by the FFT filtering by the processor resulting in a numeric value indicative of the presence (or not) of the 800Hz⁵ tone being transmitted by the sender. At the very end of each frame the processor determines if the threshold for a mark has been reached for the the 800Hz receive tone; otherwise a space is assumed to be present.

Hardware Implementation of a “Modern CCW Transceiver”

Thus far, I've been describing Coherent CCW principles of operation from a higher-level perspective, which is good to establish an understanding of “The Big Picture”. This also provides a good foundation for understanding the specific implementation taken for the “modern” CCW transceiver being presenting in this paper.

² The CCW Archives – <https://midnightdesignsolutions.com/ccw>

The specific approach taken with this design was to utilize the K1SWL “PMK” (PicoMite Memory Keyer) both as a base **computing platform** as well a physical **base hardware platform**.

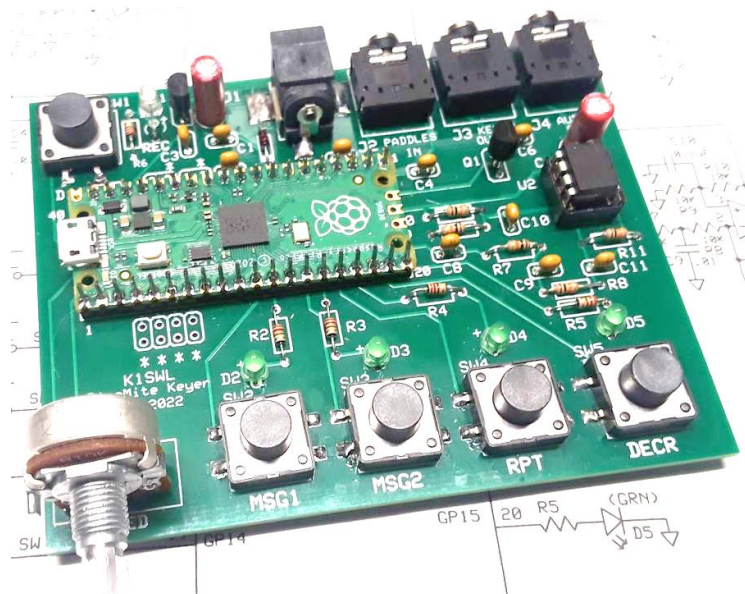


Figure 3: K1SWL’s “PicoMite Memory Keyer”

As a **computing platform**, the PMK uses the Raspberry Pi Pico as the microcontroller for all of Benson’s Keyer software, yet it still has plenty of room and horsepower left over for additional capabilities such as CCW processing. The Pico processor is amazingly powerful for the price of about 5 dollars; and amazingly flexible with 26 multi-function I/O ports that can be programmed for special purposes such as interrupts, i2c/spi/rs232/usb communications, ADC, PWM, timers, accelerated on-chip floating point libraries, and more. The Pico also sports the popular RP2040 Arm Cortex M0+ processor operating at 255 MHz with two cores that may be independently programmed to work interactively on given software applications. It’s no small wonder why K1SWL chose to experiment with this microcontroller when it was introduced some eighteen months ago. More about the Pi Pico can be found online (see [References](#) section).

As a **base hardware platform**, the PMK design was serendipitously timed with the genesis of ideas the authors had for developing a modern approach to a CCW transceiver. I had lightly collaborated with Dave during his own development of the PMK and it became evident to me that not only would the Pico microcontroller be “the one for me” too but his Keyer PCB would serve as a superb **base** computing and I/O platform for additional purposes. Since the 40-pin Pico device package has all signals readily available on its forty pins, a *scathingly brilliant*³ idea arose to develop a daughtercard, or “shield”, to plug directly onto those pins, essentially extending the capabilities of the PMK base board. Further, the overall CCW ‘system’ then has full use of all the switches, LEDs and I/O jacks available on the baseboard. Such a deal!

Using this technique, I designed a general purpose PMK shield for hams containing an OLED display, a GPS receiver module and an Si5351 clock generator module that fits in place above the Pico using stackable pinheaders – double-ended long pinheaders on the Pico, which then (usually)

³ Spoken with a mock British accent, “Scathingly brilliant” was a term often used by my mentor Joe Everhart N2CX, as derived from the 1966 movie “The Trouble With Angels” starring Hayley Mills playing twins in an orphanage. <https://www.imdb.com/title/tt0061122/characters/nm0001539>

plug into SIP sockets on the PMK board. Dual-20 pin SIP sockets are also soldered on the *bottom of the Shield* which allows it to plug onto the tall pinheaders extending up from the Pico. The illustration below should make this arrangement clearer!

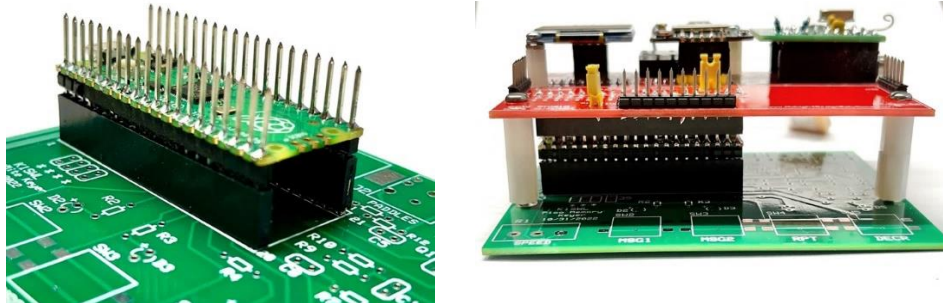


Figure 4a: (LEFT) Double-ended, long pinheaders attach Pico to the PMK. These pinheaders allows the Pico to plug into a set of SIP sockets on the PMK pcb. (It's a good practice to use sockets for programmable devices.)

Figure 4b: (RIGHT) Shield plugs onto the Pico's pinheaders. Photo shows how the CCW Shield plugs onto the top of the Pico's long pinheaders to form a compact and solid assembly, while also providing access to every signal that the Pico has to offer.

To complete the hardware picture, Phase I of the CCW Transceiver interfaces with an external transceiver using the PMK's "Keyer Out" line to key the transceiver and an analog input from radio's audio for signal decoding.

This Shield may be used by anyone wishing to further extend the Pico or CCW capabilities, or even to develop custom projects based on this platform. Phase II of the CCW project adds the RF deck to the Shield, as covered in Part 2 of this article series, thus customizing it as a CCW Shield (or generally, a *transceiver shield*).

Two more notes about the hardware. The first is in regard to the schematic (see the Schematic at the end of the paper) – all components on the lower half are on the completely unmodified PMK baseboard. Other than use of the stackable headers on the Pico processor, *no hardware changes are required of the off-the-shelf Pico Memory Keyer kit*. Further, the main interface between the baseboard and Shield is shown as a horizontal dashed line J5 between the two boards. This demarcation interface makes available all the Pico signals to the components on the shield board plugged into it, making it convenient for developers looking to extend/customize board capabilities.

So, the simplicity of the Shield is evident in the schematic ... there's not much there! The OLED provides concise display of system function, the GPS module provides precise and accurate time for QSO synchronization, and the Si5351 clock module serves as the receiver local oscillator and main transmitter frequencies. Simple hardware modules ... and the rest is "just" in the software.

The Software "Big Picture"

While the hardware was relatively simple and straightforward, the software design proved to be the largest challenge in this project. In retrospect, it's hard to believe that it's taken almost a year to get to this point of demonstrated "phase I" CCW capability; when writing this article it seems like the solutions were obvious all along. Perhaps it is often this way or maybe it just takes me a

long time for me to wake-up⁴. // Also, being retired from the day job it's easier these days to stop to watch a movie, sleep in, and get distracted with myriad other shiny object projects along the way! Perhaps readers can identify with this syndrome.

The software part of this Modern CCW Transceiver project consists of three main efforts:

- 1) GPS Time Synchronization
- 2) Paddle Detection and Time-sync'd Transmit of Dots & Dashes
- 3) Decoding the Received Bitstream

1) GPS Time Synchronization

At the heart of this, and earlier, CCW systems is a stable and accurate 10-Hertz clock that provides 100-millisecond “frames” synchronizing both the transmit and receive stations. In the old days this way was accomplished as best as possible by carefully synchronizing to WWV for correct time (to seconds accuracy!) and using a very stable, low-drift local master oscillator derived from the radio being used. This sound complex – and it was indeed difficult to achieve the stability required for CCW use.

Today we have the luxury of having an elaborate network of GPS satellites orbiting the Earth, providing fairly accurate time and clock signals for all sorts of uses in daily life. In this project we make use of this by employing a small and inexpensive GPS receiver board that, once connected and “locked” with the satellite constellation, is able to provide both the absolute time and accurate, cesium-based clock signals. I used the uBlox **u-Center** application to pre-configure each GPS receiver card to provide a 10Hz clock signal from its time pulse output pin, which connects to one of the input pins on the Pico.

This pin generates an interrupt every 100ms (i.e., the period of a 10Hz clock signal) which the CCW software uses to determine the “Start Of Frame”. The ISR (Interrupt Service Routine) is pretty neat whereby all it does is quickly set a flag whenever the SoF is detected; and when the Pico is processing paddle inputs for transmit, or when it is looking to decode incoming data, the SoF flag is monitored and action is taken when it sees it change state. The use of this flag is integral to CCW operation and will be discussed in more detail shortly.

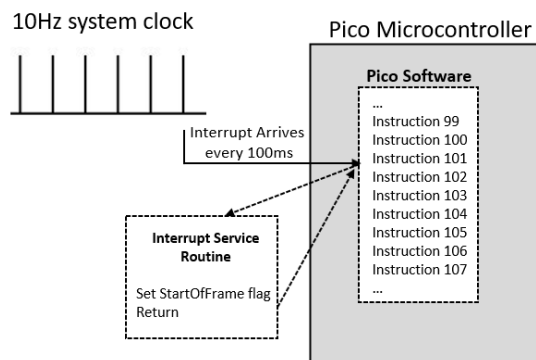


Figure 5: 10HZ clock interrupts the Pico every 100ms. This is the main heartbeat of the CCW system.

⁴ "Problems are nothing but wake-up calls for creativity" – Gerhard Gschwandtner

2) Paddle Detection and Time-sync'd Transmit of Dots & Dashes

This is the biggie of the Software Big Picture. The CCW specification states that Morse code must be sent on explicit time boundaries so the receive side knows when to start and stop decoding for each possible code element being received. This is the *phasing* part of “knowing three things about the transmission”, mentioned at the top of the paper. Those time boundaries are 100ms periods (frames) set by the 10Hz master GPS clock, and they define the Morse code speed of 12 wpm; so it is imperative that the code being sent by hand be matched up to the precisely-defined 100ms frames.

The challenge for this effort is that code sent by humans using paddles or a hand key is completely *asynchronous* in nature – code sent by the operator occurs at random times relative to GPS time, which is what CCW necessitates. It is nearly impossible to send perfectly-spaced code, even for the best of fists using the most accurate electronic keyer, because none is clocked by GPS-derived signals. Further “perfect” code – that which is sent with exact 3:1 dot-to-dash ratio – sounds quite boring and staccato-like, so most hams adjust the ratio weighting to give their fist a little personality. Hams using mechanical bugs personify this personality trait and certainly don't come close to CCW speed standards!

The solution I selected was to use the time-tested D-flipflop logic element to sync an event to an external clock. The idea is that when an asynchronous event like a paddle contact closure occurs, the logic level is placed on a flipflop's input for Data and the logic block waits for a clock to come along pass it on through to the Q output. We can do this in software by hooking (intercepting) Dave's software when the DIT or DAH bit is set and then sit waiting for the next 100ms frame to come along before acting on it.

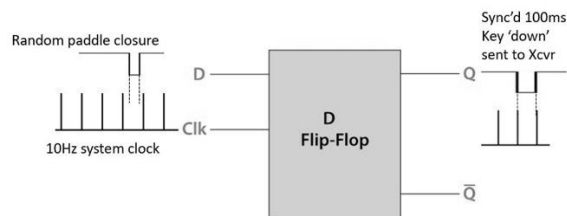


Figure 6: Simple representation of a D-flipflop

Simple and elegant, right? Consider the following rather complex-looking illustration of a stream of 100ms frames and the sample closure of first a dit (at 'A') and then a dah (at 'B'). First a dit occurs someplace in the middle of a 100ms period and we need to remember it (with a D-flipflop flag called “DitLatch”) so we can key the transmitter at the start of the very next frame. We also must remember that the bit was set at the start of the next frame so we can turn off the key. Dit processing is pretty easy.

Handling the dahs is slightly more complicated. Recalling that dahs are three times longer than dits, we need to keep the transmitter keyed for three frames. So in addition to a DahLatch to tell us to go “key-down” at the start of the very next frame, we need a DahCntr to count how many frames to keep the key down. Still pretty easy.

The complexity lies in the fact that both the 'A' and 'B' processing scenarios **occur at every single 100ms frame boundary** when the user is sending. Further, in Phase I of the software I am restricting paddle use to a single-lever version – not dual-lever “squeeze keying” type – in order

to reduce probable complexities arising from the timing of two contact closures within the same window, or the logic labyrinth of more advanced keying styles (e.g., Iambic A/B, Ultimatic, *et al*).

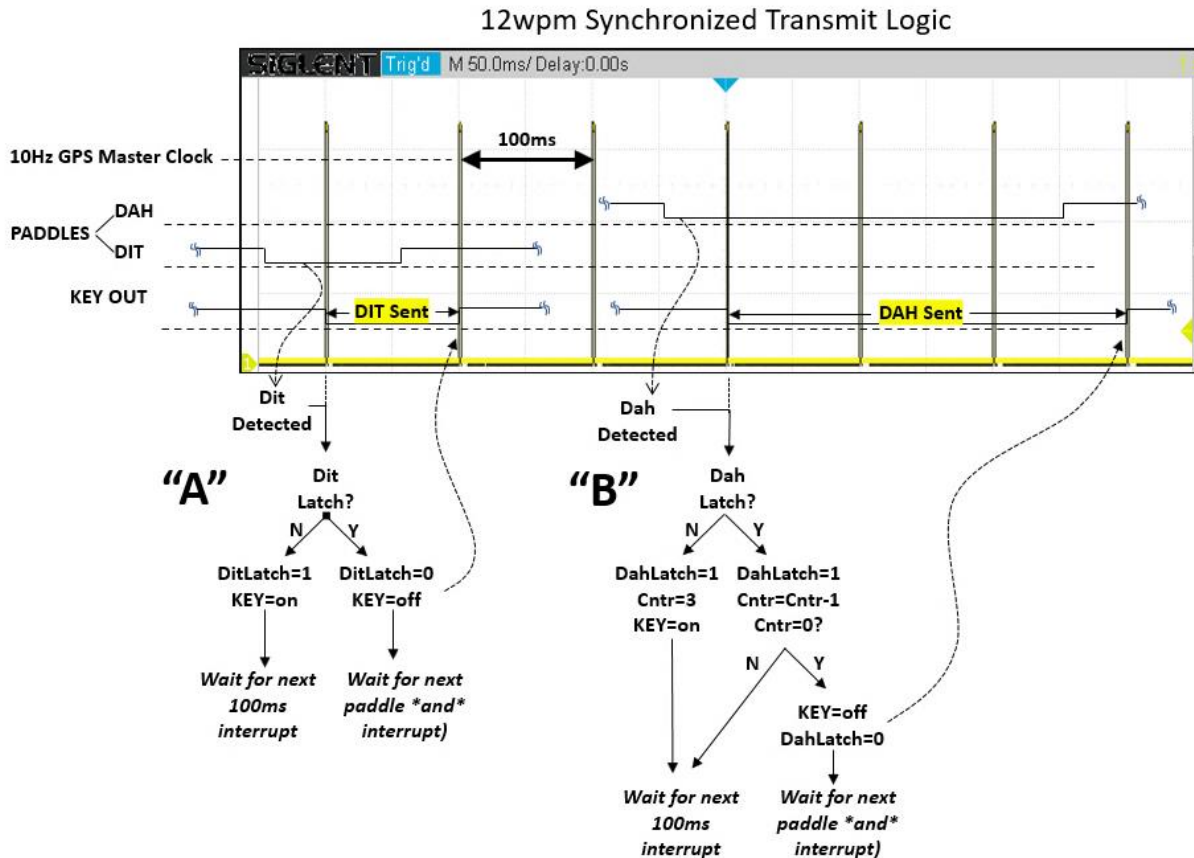


Figure 7: 12wpm synchronized transmit logic

3) Decoding the Received Bitstream

This last of the three main efforts in the project was the trickiest wherein DSP techniques were employed within each of the 100ms frames to determine if an 800Hz tone⁵ was being received, as sent by the CCW transmitting station. Yet curiously, this is the easiest effort to explain! I needed to study and experiment a lot to understand the sampling mechanism with the Pico’s Analog-to-Digital Converter (ADC), and especially for use of its accelerated Fast Fourier Transform (FFT) computations, but once I had this somewhat under my belt it was a straightforward move to implementation. A handful of charts and illustrations tell the story better than words, so here we go. Please read the captions for each graphic in the order presented.

⁵ Most ham transceivers use an 800Hz transmit or receive offset when in CW mode, thus delivering an audio tone of that frequency to the receiving station’s speaker.

Sampling & FFT Information	
ADC depth	12 bits
Fs (max 500 kS/sec)	4000 Hz
Samples	255
Bin Res (Fs/Samples)	15.69 Hz
Window (measured)	95 ms
Bin (of 800 Hz tone)	51.00
FFT Magnitude	(See display & data tables)

Figure 8: Sampling & FFT Information. The chart shows that with a sampling frequency of 4000Hz (usable to only 2000Hz before aliasing occurs) and 256 samples (127 samples are usable after the FFT is performed, in pairs), we end with having a “Frequency Resolution” (or filter bandwidth) of 15.7Hz. This is not as narrow as the 10Hz width of Petit’s original CCW system, but it’s pretty darned close, and still something far, far narrower than what could be achieved with electronics.

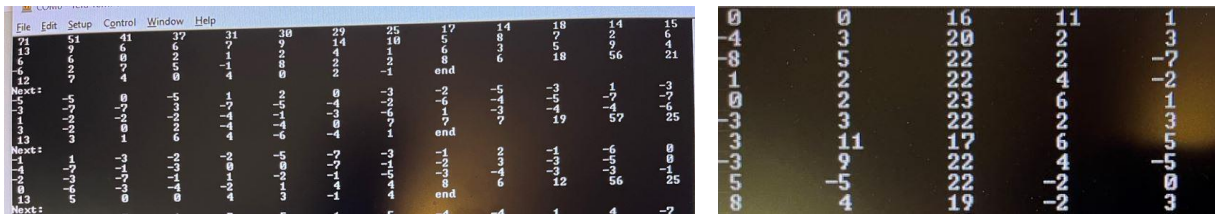


Figure 9a: (LEFT) FFT bin counts for bins 0-59. Screen shot of FFT results on actual data collected during reception. Note the “sweet spot” surrounding bin 51 representing 800Hz. Note the big principle enabling sampling and successful representation of a single frequency: “Zero-bias for noise, Signal for a constant signal”. What this means is that the average signal in the presence of noise will tend toward a zero value, while that of a signal-of-interest (800Hz in our case) will skyrocket out of the noise and be readily apparent to subsequent processing.

Figure 9b: (RIGHT) FFT bin counts for sweet-spot: bins 48-52. This screen shot shows that if we look at the bins in/around those for 800Hz we see a clear and distinctive peak, so we can be relatively assured that a CCW signal is being received. *This is why a CCW can be picked out of a “dead channel” as much as it is able to be spotted amongst a crowded band during CQWW contesting weekend or even in the FT8 band!* Nobody but the CCW-transmitting station has the characteristic of sending the 800Hz tones for properly formatted Morse code; hence only the CCW station will come through loud and clear. Man, what a filter!

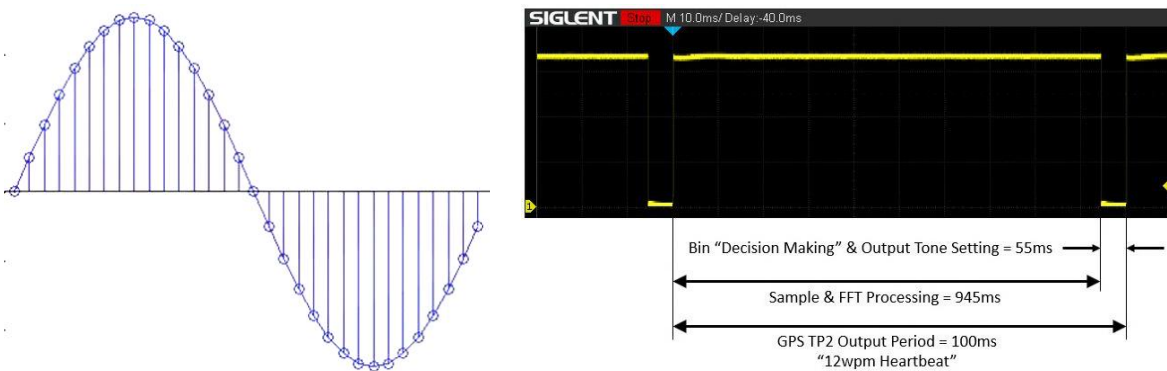


Figure 10a: (LEFT) Discrete sampling of a waveform performed by the Pico ADC. The software then takes the 255 samples and performs the FFT “in real time”, which all happens within a 945ms window.

Figure 10b: (RIGHT) 55-milliseconds remain for mark/space decision making. The 945ms sampling and FFT computation time is illustrated in this annotated oscilloscope trace. Note that the software has a whopping 55ms remaining in the frame to make a decision of whether the received data is a mark or a space – *Plenty of time!*

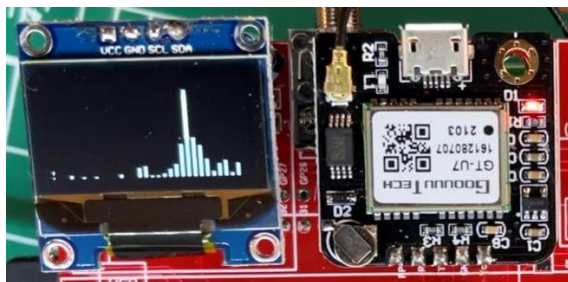


Figure 11: Frequency domain representation of the received signal after sampling and FFT processing. Note the frequency peaking at 800Hz ... How do we know the peak is at 800Hz? It's at the 51st bin, which at 15.7Hz/bin yields the expected 800Hz value, which is also the value of the receive tone coming from the speaker. (Using the SPOT pushbutton on the Elecraft K3 was a joy during development since it auto-adjusted the receiver dial to yield the 800Hz receive tone!)

Two Final Notes

Not much has been mentioned yet about how the **Date/Time reporting capabilities of the GPS receiver** are used. Although somewhat obvious and relatively straightforward, they nonetheless play a very important role within the CCW system. I initially mentioned that three factors are required for successful CCW communications, and knowing exactly *when* to start the 100ms framing windows is an essential one of the three. The CCW systems of the early days needed to adjust the *phasing* of the receive-side 10Hz master clock relative to the timing of the incoming bit stream to peak reception signals. This was accomplished by sliding the local clock relative to the incoming data stream to maximize frame overlap and ensure synchronization. However in today's system, with both the transmitter and receiver endpoints starting their respective 10Hz clocks at the very same time, there is no need for sliding windows or phase adjustment! When a CCW station operator presses a button on the PMK to "Start CCW Transmit", he/she can be assured that any other CCW station listening on that exact frequency will be able to copy the Morse code transmission because of the common use of GPS timing. // *Time will tell (ha ha) if something is yet needed to account for propagation path delays on longer-haul connections.*

Also not mentioned *at all* thus far is the **software language** used for developing this modern version of a CCW transceiver system. The Pi Pico proves to be an amazingly flexible platform by allowing developers to program in a number of different languages: Arduino C, C++, MicroPython, CircuitPython, Assembly Language, Scratch, HTML5, JavaScript, JQuery, Java, Perl, Erlang ... and even in Basic! If your eyes have glossed over on this list of languages, you're not alone, but this gives you some idea of the ubiquity of application that this marvel has. And in most cases all one needs to do is plug a brand new Pico into your computer's USB port and drop a special file into the folder that appears in your finder window, essentially outfitting the Pico with the required drivers and native library calls for that language to operate the microcontroller. Pretty amazing all for \$5.

So to answer the question of what language is being used for the PMK and the CCW project ... <drumroll, please> ... it's done in Basic! "No way!" Yes way, and in fact the version of Basic Dave used for the PMK and that I also adopted for developing the CCW program is called **MMBasic**, optimized and enhanced for the Pico by Geoff Graham. Take a look at the source code for both the PMK and the CCW project, located on the PMK web page – we think you'll be impressed with the utility and performance of the Pico under control of *this language of the past driving some projects of the future!*

Next Time

In Part 2 of this epic article, we discuss our adventures with two-way CCW QSOs between principal parties: N2APB, WB9FLW and K1SWL. Also in process are refinements to the decoding algorithms, connection and synchronization protocols, increases in keying speeds, and additional features for the user interface. Lastly, an exciting parallel project under development will be presented: an RF deck that plugs directly onto the CCW Shield thus achieving the original goal of making this project a completely standalone CCW Transceiver.

The Authors

George Heron, N2APB ... Licensed more than fifty years, George is now retired from EE and cyber security careers and is living in rural TN. Always involved in technology and homebrewing, he helped to form and lead the NJQRP, the AmQRP, and Chat With the Designers; he edited and published QRP Homebrewer and Homebrewer magazines; and he co-designed numerous projects under the Midnight Design Solutions shingle including the SDR Cube and NUE-PSK transceivers, DDS-60 siggen, Midnight Ultimate Keyer and Precision Clock kits, and the Rainbow Tuner. George may be contacted at n2apb@MidnightDesignSolutions.com.

Peter Eaton, WB9FLW ... Pete was first licensed as a Novice in 1963 (WN3EJB), then as a General (WA3KZA) and Advanced (WB9FLW and KA2PE) while stationed at Yokota AFB Japan. He was very active in TAPR during the early years and wore many different hats which included Executive Vice President, speaker on Packet Radio at many conventions, and was part of the Team that design three generations of Terminal Node Controllers. His current interests are Weak Signal Modes (thus CCW). Peter may be contacted at zx97lite@yahoo.com.

References:

- 1) **PicoMite Memory Keyer**, by Dave Benson K1SWL – <https://www.midnightdesignsolutions.com/pmk/>
- 2) **Raspberry Pi Pico microcontroller** – <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>
- 3) **Coherent CW Archives** ... Unofficial archive of the works from the early pioneers of Amateur Radio's digital communications era. <https://midnightdesignsolutions.com/ccw>
- 4) **MMBasic** ... <https://geoffg.net/MaximiteBasic.html>

Schematic

