Introduction

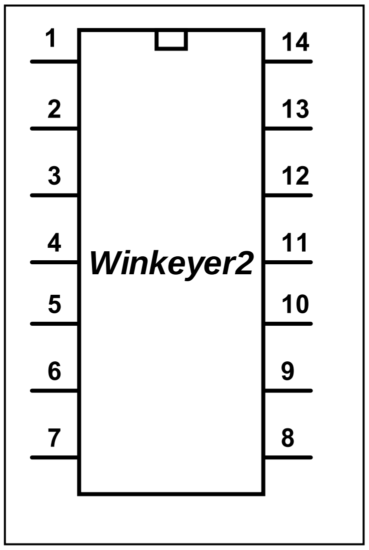
This document describes the Winkeyer2 IC and its command interface. Winkeyer2 is a second generation single chip Morse keyer IC. It is designed to attach to a PC's serial port and provide accurate transmitter keying for a Windows based logging or other ham radio software package. Due to timing latency inherent in the multi-threaded Windows operating system, it is difficult to generate accurately timed Morse. Winkeyer2 buffers ASCII characters sent by a Windows based software application. It then translates them to Morse and directly keys a transmitter or transceiver. In addition, Winkeyer2 has paddle inputs so that an operator can break-in and send using paddles at any time. Winkeyer2 also provides a speed potentiometer interface so that an operator can instantly dial any speed desired.

The host PC communicates to Winkeyer2 over a simple serial interface which can be a COM or USB port. Letters to send, along with operational commands, are sent from the host to Winkeyer2 over the serial link. A substantial feature list is provided allowing the user to precisely tailor Winkeyer2's keying characteristics to a particular transmitter. Winkeyer2 has a very low power requirement; in fact, it was originally designed to be powered from a PC's serial port. In standby it draws under a micro amp.

K1EL sells several kits that utilize the Winkeyer2 IC, refer to the k1el website: www.k1el.com for further information. One popular kit is WK_USB which includes a USB interface, speed control, enclosure with pushbuttons, and battery pack for standalone use.

Features

- 1200/9600 Baud Serial Rx/Tx Interface
- Iambic CW Paddle Interface
- Two Key Output Ports (high true TTL)
- Two PTT Output Ports: (high true TTL)
- 25 ma output sink/source
- Adjustable PTT lead in and tail delays
- Adjustable Speed 5-99 WPM
- Adjustable Weighting and dit/dah ratio
- Adjustable Farnsworth Character Spacing
- Adjustable Keying Compensation
- Autospace and Letterspace Control
- Sidetone Output
- Standalone Enhanced K12 Mode
- Paddle only sidetone
- Dit/Dah Memory Control
- Adjustable First Dit/Dah correction
- Adjustable Paddle Switchpoint
- Iambic A, B, ultimatic & "Bug" modes
- Speed Pot Interface
- Adjustable speed pot range
- Embedded commands
- 128 character input buffer
- No crystals or oscillators
- Single 5-volt operation
- Current Draw: < 2 ma in active operation
- HSCW and QRSS Capability
- Power Down Sleep
- Six stackable memory slots



Pin 1 – Vcc (5.0 volts)
Pin 2 – Port 2 Key Output
Pin 3 – Port 2 PTT Output
Pin 4 – USB Connected Sense
Pin 5 – Serial Receive Input
Pin 6 – Serial Transmit Output
Pin 7 – Port 1 Key Output
Pin 8 – Sidetone
Pin 9 – Port 1 PTT Output
Pin 10 – Speed Pot Analog Input
Pin 11 – Right Paddle Input
Pin 12 – Left Paddle Input
Pin 13 – Switch Array Input
Pin 14 – Vss (Ground)

Figure 1 – Winkeyer2 Package & Pinout

Theory of Operation

This section will describe how the Winkeyer2 works. As shown in Figure 1, the host PC is connected to Winkeyer2 over a serial COM port, which can be a USB port supporting virtual COM. Winkeyer2 is a slave to the PC in that it receives commands and data from the PC. The PC can send commands while Winkeyer2 is sending Morse allowing dynamic configuration changes. Winkeyer2 will communicate back to the host for four reasons:

1) Inform the host of a status change in Winkeyer2.
2) Inform the host of a speed pot or pushbutton change (WK2 mode only).
3) Respond to a request for information from the host.
4) Echo back Morse in ASCII as it's being sent from either the serial port or the paddles.

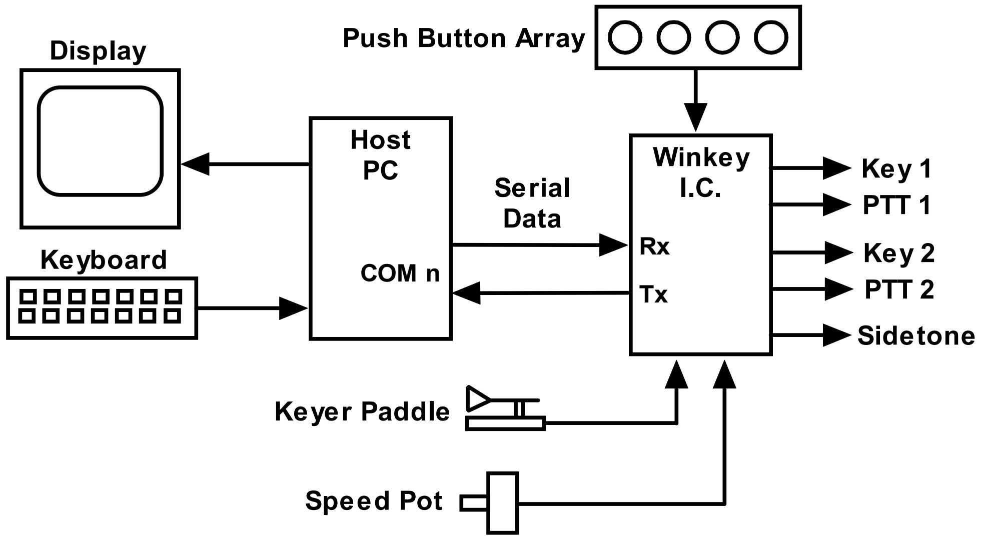


Figure 2 – Winkeyer2 to PC Connection

There are two types of serial input from the host to Winkeyer2: Command and Data. Commands modify Winkeyer2's operation in some way, for example changing operating speed, pausing transmission, or asking for status. Data can be letters, numbers, or prosigns that are to be sent in Morse. Commands and data are processed differently in Winkeyer2. Data is put into a serial buffer that allows the host to send data ahead of the Morse being sent. The size of this buffer is 128 characters and is a FIFO which is an acronym for First In First Out. This means that characters are taken out in the order they were put in. Since there can be a considerable delay from host input to Morse output, commands bypass the input FIFO and are acted upon immediately. This allows changes to be made while sending is underway.

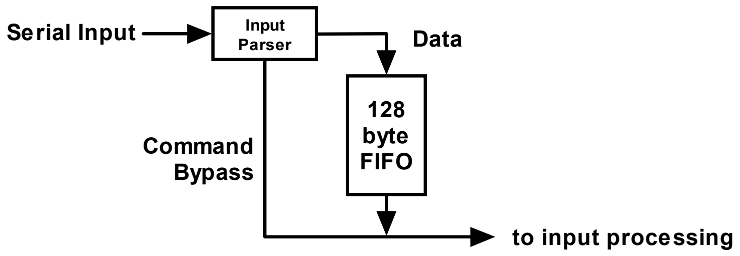


Figure 3 – Data and Command Flow inside Winkeyer2

Since there are times when you don't want commands to take effect immediately, Winkeyer2 allows commands to be buffered. This means that the command is placed in the serial buffer and won't be acted on until it comes out of the buffer. An example of the use of a buffered command would be to send two words at two different speeds, the first at 15 WPM and the second at 20 WPM. By placing a buffered speed command

between the words the speed will not be changed until the first word is completely sent. Not all, but many of the immediate commands can be entered as buffered commands.
Communication from Winkeyer2 to the host operates in a loosely coupled manner. This means that the host never issues a command and waits for a response. Instead, the host sends a request for information and Winkeyer2 queues this request and will respond to the host when processing time allows. Winkeyer2 processes tasks in parallel, there may be other bytes waiting to be sent back to the host before the latest request can be handled. Rather than wait for a return, the host should divide its Winkeyer2 driver interface into two parts, one part that issues command bytes and a second part that checks for returned bytes and processes them when they arrive. Following is a bit of pseudo-code that illustrates this concept. It will make more sense as you learn about the Winkeyer2 command set.

```
Serial Comm Thread {
  while (1) {
        if (host has a command to send to Winkeyer2) {
                send command to winkeyer2;
        }
        else if (Winkeyer2:uart_byte_ready) {
                wkbyte = Winkeyer2:uart_read();
                if (( wkbyte & 0xc0) == 0xc0 {
                        it’s a status byte.  (Host may or may not have asked for it.)
                        process status change, note that it could be a pushbutton change
                }
                else if ((wkbyte & 0xc0) == 0x80) {
                        it’s a speed pot byte (Host may or may not have asked for it.)
                        process speed pot change
                }
                else {
                        it must be an echo back byte
                        if (break-in==1) { it’s a paddle echo }
                        else { it’s a serial echo }
                }
        }
  }
}
```

Notice that unless Winkeyer2 has something for the host to read, the host continues to process outgoing commands and other tasks. Also note that speed pot and status bytes can be unsolicited, in other words Winkeyer2 can send these at any time a state change occurs inside Winkeyer2. Echo back bytes are also unsolicited as they are based on asynchronous Morse sending. The host has to be able to handle these as they occur. If host processing is slow, a serial input buffer on the host side is required to make sure no returned bytes are missed.

Paddle Input Priority

Winkeyer2 accepts input from either its serial port or iambic paddle port. Paddle input will always take priority and will interrupt serial data, automatically clearing Winkeyer2’s serial input buffer. When a paddle break-in occurs, any additional serial data that arrives from the host will be processed, but will be ignored unless it is an immediate command. After paddling ceases, Winkeyer2 will pause for one word space time before it resumes serial data transmission.

Standalone Keyer Mode

Winkeyer2’s primary purpose is to provide accurate Morse keying to a Windows based application. The most often requested feature from users of WK1 was to also allow WK to be run by itself, not connected to a computer. In response, a complete standalone keyer has been included in Winkeyer2.

In most respects the standalone keyer is completely separate from the host mode keyer. This means that the standalone keyer has its own configured state that is preserved when changing to host driven mode. Configuration changes issued while under host mode will be cancelled when returning to standalone mode. A simple Windows application is available from K1EL called WK2MGR which can be used to set the standalone configuration and message contents from the PC. In addition settings and messages can be entered by paddle commands in standalone mode.

Winkeyer2 standalone mode is an emulation of the popular K1EL K12 keyer IC command set. Several enhancements are included as documented in the Standalone keyer section.

Power Up Default State

On power up, Winkeyer2 comes up in standalone mode and stays in that mode until it receives a Host Open command from a PC host. At that time standalone mode is suspended. When the host takes over it normally downloads a block of initialization parameters (see Load Defaults command) to set the operating state as desired and to sync Winkeyer2 settings with those of the host. All applications interfacing to WK2 should always issue an Admin:Close command upon application shutdown to return WK2 back to standalone mode. This allows the keyer to be used in standalone mode while still attached to the PC. When WK2 is physically disconnected from the host it automatically goes into standalone mode even if a Close command was not issued.

While WK2 is attached to a PC com port in the closed state it will accept ADMIN commands. This allows an application to set WK2/WK1 mode, upload or download standalone messages and standalone settings.

Winkey Lockup Recovery

Once Winkeyer2 is connected to a host it should not be physically disconnected while the host application is active. Accidents do happen and if the USB cable is pulled in the middle of a command or data exchange Winkeyer2 can get locked up. It's not very likely but it can happen. A provision is included to easily get Winkeyer2 back in operation again. Press and hold the command button until Winkeyer2 responds with a restart response. If Winkeyer2 was stuck in host open a **C** will be sent in sidetone. If Winkeyer2 was locked due to a different problem (very unusual) it will echo an **R**.

Pushbutton Notification

Winkeyer2 supports pushbutton inputs that are intended for standalone operation. One pushbutton is designated as the Command push button and is used to initiate paddle commands. The other pushbuttons are used to enter or play recorded Morse messages. A provision has been included that will notify the host when there is a change in pushbutton state. To enable this WK2 feature, the host application must issue an Admin:Set WK2 Mode command. Once the mode is sent, pushbutton status will be returned in the WK2 status byte. Standalone operation is unaffected by Winkey mode setting.

USB Sense

Winkeyer2 was designed with USB interfacing in mind. A USB sense input is provided which should be asserted high when the USB port is attached and active. If the USB port is switched to standby or disconnected from the PC, USB Sense should be pulled low. When low, WK2 is allowed to go into low power sleep mode. Note that while connected to the host with USB sense high, WK2 will not go into low power standby.

Version 22 was changed to handle the case where the host PC goes into standby while a host application is actively connected to Winkeyer2. Winkeyer2 will not disconnect and remain in host mode until the PC comes out of standby. This preserves the application to WK2 link even if the PC goes into standby unexpectedly.

Host Mode Command Descriptions

This section documents the commands that are sent from the host to Winkeyer2 over the serial interface. Commands are special hex codes that are sent to Winkeyer2. These codes range from 0x01 through 0x1F. In this document a hex value will be presented in angle brackets, for example **`<02>`**. Some commands have one or more parameters sent immediately after the command code is sent, this will be documented as the command code followed by a value: **`<02><nn>`** where nn is a single byte binary value. The notation **[c]** represents a single ASCII character sent to Winkeyer2 as a single serial byte.

Immediate Commands

These commands are processed as soon as they are received, they bypass the input buffer.

● Admin Command <00><nn> nn is a value from 0 to 20

After power-up the host interface is closed, serial status, echo, or pot change data will be not be sent to the host. The only commands WK2 will accept are Admin commands. Admin commands are received, processed and any return status or data will be sent back immediately. Admin commands calibrate the interface, reset WK, obtain debug information and open the interface. With the exception of the Amin:Close command, all Admin commands should only be issued while the host interface is closed. Following are descriptions of the Admin commands:

0: Calibrate	This is an historical command preserved for WK1 compatibility. It is no longer required for the more accurate PIC the WK2 is implemented on. You can issue the command and it will not cause ill effects, but it is not processed by WK2. The command syntax is: <00><00> pause 100 mSec <FF>
1: Reset	Resets the Winkeyer2 processor to the power up state. Do not send this as part of the initialization sequence. Only send this if you want to do a cold reboot of WK2.
2: Host Open	Upon power-up, Winkeyer2 initializes with the host mode turned off. To enable host mode, the PC host must issue the admin:open command. Upon open, Winkeyer2 will respond by sending the revision code back to the host. The host must wait for this return code before any other commands or data can be sent to Winkeyer2. Upon open, WK1 mode is set.
3: Host Close	Use this command to turn off the host interface. Winkeyer2 will return to standby mode after this command is issued. Standby settings will be restored.
4: Echo Test	Used to test the serial interface. The next character sent to Winkeyer2 after this command will be echoed back to the host.
5: Paddle A2D	Historical command not supported in WK2, always returns 0.
6: Speed A2D	Historical command not supported in WK2, always returns 0.
7: Get Values	Returns all of the internal setup parameters. They are sent back in the same order as issued by the Load Defaults command. Again, this command is a diagnostic aid. Only issue this command when host interface is closed.
8: Reserved	***K1EL Debug use only***
9: Get Cal	Historical command not supported in WK2, always returns 0.
10: Set WK1 Mode	Disables pushbutton reporting
11: Set WK2 Mode	Enables pushbutton reporting, alternate WK status mode is selected.
12: Dump EEPROM	Dumps all 256 bytes of WK2's internal EEPROM.
13: Load EEPROM	Download all 256 bytes of WK2's internal EEPROM.

14: Send Standalone Message Command WK2 to send one of its internal messages. The command syntax is: <00><14><msg number> where number is 1 through 6

15: Load XMODE Load mode extension register.

Value nn Bits	Function
7 (MSB)	Standalone cut mode, send 0 as T and 9 as N when set to 1 (ignored in host mode)
6	Unassigned
5	Enable Paddle Status when = 1
4	Unassigned
3-0	Letterspace Adjustment, lower 4 bits 0 to 15 in 2% increments

Table 1 – Mode Extension Register

16: Reserved If this command is issued, WK2 will respond with a zero.

17: Set High Baud Change Baud Rate to 9600 baud

18: Set Low Baud Change Baud Rate to 1200 (default)

19: Reserved Function TBA

20: Reserved Currently unassigned

Baud rate change must be handled in a specific way. Since most applications expect WK2 run at 1200 baud, this is always the default and will be reinstated whenever WK2 is closed. If an application wants to run at 9600 baud, it must start out at 1200 baud mode and then issue the Set High Baud command. When the application closes it should issue a WK close command which will reset the baud rate to 1200.

● Sidetone Control <01><nn> nn is a value determined from Table 1 & 2

In WK2 sidetone is always enabled and pin 8 functions as the sidetone square wave output. The following tables define the format of value **nn**. *Note that these frequencies are slightly different than WK1*

Value nn Bits	Function
7 (MSB)	Enable Paddle Only Sidetone when = 1
6-4	Unused set to zero
3-0	Sidetone frequency N (See Table 2 below)

Table 2 – Sidetone Control Assignments

N	Frequency		N	Frequency
0x1	**4000 Hz**		**0x6**	**666 Hz**
0x2	**2000 Hz**		**0x7**	**571 Hz**
0x3	**1333 Hz**		**0x8**	**500 Hz**
0x4	**1000 Hz**		**0x9**	**444 Hz**
0x5	**800 Hz**		**0xa**	**400 Hz**

Table 3 – Sidetone Selection Table

The most significant bit of the frequency byte controls the paddle only sidetone feature. In WK2 you can choose to only use sidetone for paddle entry and mute it for CW sourced from the host port. This is called Paddle Only Sidetone and is selected by setting the MSB of the sidetone control value.

● Set WPM Speed <02><nn> nn is in the range of 5-99 WPM

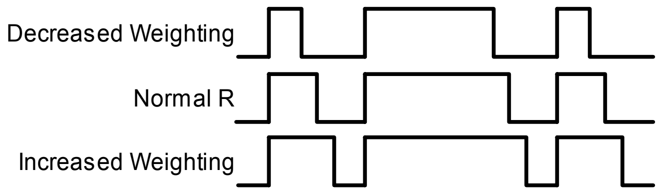
Example: <02><12> set 18 WPM

Set a new Morse operating speed, this command takes effect as soon as Winkeyer2 receives it. If speed is set to zero then Winkeyer2 will take its speed setting directly from the speed pot., this is the reset default.

● Set Weighting <03><nn> nn is in the range of 10-90%

Example: <03><32> for weight=50

This command allows a proportional amount to be either added or subtracted from the length of all dits and dahs sent. A value of 50 (0x32) selects no weighting adjustment. Values less than 50 reduce weighting and values greater than 50 increase weighting. Note that weighting does not affect sending speed because any increase in keyed time is subtracted from spacing time. Reduction in weighting results in a thinner sounding keying while increased weighting results in a heavier sound. Since weighting tracks speed, a given weighting will sound the same at all speeds.



WEIGHTED DIT TIME = (((WEIGHT×50)/50)×DIT TIME)

WEIGHTED DAH TIME = (((WEIGHT×50)/50)×DAH TIME)

Figure 4 - Weighting Example

● Set PTT Lead/Tail <04><nn1><nn2> nn1=lead in time, nn2=tail time

Example: <04><01><A0> lead-in=1, tail=160
Values can be 0 to 250 in 10 mSecs steps

Winkeyer2 provides a transmitter PTT output for each key output that can be used to switch a transmitter or linear amplifier over to transmit mode in advance of actual CW keying. You have control over the time delay between when PTT is asserted and when CW keying will start, this is lead-in. You also have control over how long the transmitter will stay in transmit after keying has stopped. These delay settings apply to both key ports. The trailing delay is handled differently for CW sent by paddle and CW sent by "machine". Paddle delay is controlled by the Hang Time setting in the PINCFG register (see the *SetPinConfig* command). The Tail setting determines the delay used for CW sent by an internal message or CW sent by a Host application. The formula to calculate tail time is:

```
Tail Delay = Three Dit Times + (Tail Setting times 10 milliseconds)

    Examples:
    At 20 WPM, Tail set to 7, Tail Delay = (3x60)+(7x10) = 250 mSec
    At 40 WPM, Tail set to 7, Tail Delay = (3x30)+(7x10) = 160 mSec
    At 20 WPM, Tail set to 0, Tail Delay = (3x60)+(0x10) = 180 mSec
    At 15 WPM, Tail set to 55, Tail Delay = (3x80)+(55x10) = 790 mSec
```

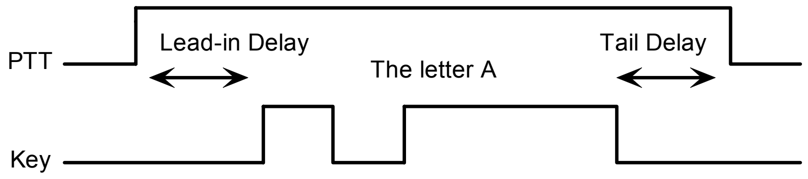


Figure 5 – PTT Lead-in and Tail Example

A detailed description of PTT functionality can be found on page 15.

● Setup Speed Pot <05><nn1><nn2><nn3> nn1 = MIN, nn2 = RANGE, nn3 = 0

This command sets the limits for the speed pot. MINWPM sets the lowest value returned; WPMRANGE indirectly specifies the maximum value returned. For example if MINWPM=10 and WPMRANGE=15, the full pot swing values, min to max, would be 10 to 25 WPM. Note that the max value is MINWPM+WPMRANGE. The value of the third parameter is not used but it must be included to maintain backward compatibility for applications supporting only WK1 keyers. Recommendation is to set this to zero but any value is accepted.

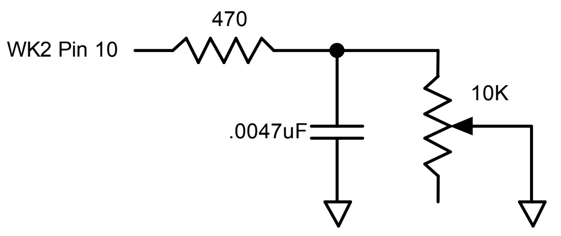


Figure 6 – Speed Pot Configuration

● Set Pause State <06><nn> nn = 01 pause, value = 00 unpause

When Winkeyer2 is paused, sending will stop immediately and will not resume until an unpause state is set. The current character being sent in Morse will be completed before pause. Note that the Clear Buffer command will cancel pause.

● Get Speed Pot <07> Return Winkeyer2's current speed pot setting.

This command will cause a speed pot command request to be queued in Winkeyer2 and it will be acted on as soon as possible. Depending on current processing load the pot status byte will be sent no longer than 200 milliseconds after command receipt. The application should not wait for a response but process the returned data in an unsolicited status handler. The returned value will range from 0 to 31 and is governed by the setting of the MINWPM and WPMRANGE values set via the POTSET command. The returned value will be the actual speed pot value minus the MIN_WPM setting. This allows the speed pot to be windowed into any 32 step range from 5 to 99 WPM. The two MS Bits of a Speed Pot status byte will always be 10:

1	0	6 bit value in WPM

Table 4 - Speed Pot Status Byte Format

● Backspace <08> Backup the input buffer pointer by one character.

This command is only meaningful if there is something in the serial input buffer, otherwise it is ignored.

● Set PinConfig <09><nn> Set the PINCFG Register

Low nibble determines how output pins are mapped
High nibble controls ultimatic mode and hang time

WK1's 8 pin package forced Pin 5 to be a shared resource, it could be assigned as a PTT output, a Sidetone output, or a secondary Key output: If it was assigned as a PTT output, that meant it was not possible to output sidetone. Likewise if it was assigned as a secondary Key output, sidetone or PTT were not allowed.

Bit 7-4	Bit 3	Bit 2	Bit 1	Bit 0
See Below	Pin5 KeyOut Enable	Pin3 KeyOut Enable	Pin 5 Sidetone Enable	Pin5 PTT Enable

Table 5 – WK1 PINCFG Format

WK2 has a dedicated sidetone pin, two PTT outputs, and two Key outputs. No sacrifices are necessary. The pin config assignments are compatible with WK1 for backwards software compatibility but allow much more flexibility than WK1. For example Bit 0 specifies whether PTT should be used, Bit 1 specifies whether Sidetone is enabled, and Bits 2 and 3 select which key port is active. If Key port 1 is active, PTT1 will be asserted in sync with KEY1 if the PTT enable is asserted, likewise for Port 2.

Bit 7-4	Bit 3	Bit 2	Bit 1	Bit 0
See Below	KeyOut 1 Enable	KeyOut 2 Enable	Sidetone Enable	PTT Enable

Table 6 – WK2 PINCFG Format

The PINCFG register is overloaded with two additional features; Dit/Dah priority and Paddle hang time. These settings are allocated to the upper four bits as follows:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3-0
Priority 1	Priority 0	Hang Time 1	Hang Time 0	See Above

Table 7 – Priority/Hang Time Format

Priority = 00 Normal Ultimatic
Priority = 01 Will send dahs when both paddles are pressed in Ultimatic mode
Priority = 10 Will send dits when both paddles are pressed in Ultimatic mode
Priority = 11 Undefined

HangTime = 0: wait 1 wordspace + 1 dit before ending paddle insertion
HangTime = 1: wait 1 wordspace + 2 dits before ending paddle insertion
HangTime = 2: wait 1 wordspace + 4 dits before ending paddle insertion
HangTime = 3: wait 1 wordspace + 8 dits before ending paddle insertion

Hang Time is like tail time, in that it holds PTT on between paddle presses, but is proportional to sending speed by virtue of the fact that it is measured in word space time while tail time is more of a fixed delay.

● Clear Buffer <0A> Reset Input Buffer Pointers

This command will reset the input buffer pointers to an empty state. It is a general clear also in that Tune and Pause are also cancelled by this command.

Clear Buffer can be sent at any time to abort a message, abort a command, or to clear the serial buffer. It will cancel any Morse character in progress immediately ending it in midstream if necessary.

● Key Immediate <0B><nn> nn = 01 key down, n = 00 key up

Use this command to implement a tune function. Once asserted, key down will remain in effect until either a key immediate with a zero value is received or the internal tune watchdog timer expires. The tune timer is hard coded to a value of 100 seconds and cannot be disabled. The key down can be aborted either by the paddles or by a clear buffer command.

● Set HSCW <0C><nn> nn = the lpm rate divided by 100

Winkeyer2 supports HSCW (High Speed CW) transmit rates through the use of this immediate command. .

For example nn=20 selects 2000 lpm and nn=35 selects 3500 lpm. Any rate from 1000 to 8000 can be picked although only a handful are actually used by radio amateurs. In the US, common rates are 1000, 2000, 4000 and 6000 lpm while in Europe 1000, 1500, 3000, 4000 lpm are common.

● Set Farns WPM <0D><nn> nn is in the range of 10-99

Example: <0D><12> for Farnsworth=18 WPM

Farnsworth spacing is useful for CW practice because it encourages you to learn characters by sound not individual dits and dahs. In Winkeyer2, Farnsworth is implemented by sending letters at a fixed rate of **nn** WPM regardless what the WPM sending rate is. Spacing between characters is determined by the sending rate. When the WPM rate is set above the Farnsworth WPM, Farnsworth is automatically disabled.

● Set Winkeyer2 Mode <0E><nn> nn = Mode bit field in binary

Example: <0E><13> set bits 4,1,0, clear the rest

The operational mode of Winkeyer2 can be modified by directly altering its internal mode register. This register is made up of eight bits which each control a particular mode.

Mode Bit	Function
7 (MSB)	Disable Paddle watchdog
6	Paddle Echoback (1=Enabled, 0=Disabled)
5	Key Mode: 00 = Iambic B 01 = Iambic A
4	10 = Ultimatic 11 = Bug Mode
3	Paddle Swap (1=Swap, 0=Normal)
2	Serial Echoback (1=Enabled, 0=Disabled)
1	Autospace (1=Enabled, 0=Disabled)
0 (LSB)	CT Spacing when=1, Normal Wordspace when=0

Table 8 – Winkeyer2 Mode Selection Table
The Winkeyer2 mode register is cleared at reset.

Bit 7
Winkeyer2 has a paddle watchdog counter that will disable the key output after 128 consecutive dits or dahs. This is to guard against the paddles being accidentally keyed continuously. By default the paddle watchdog is on but it can be turned off by setting this mode bit.

Bit 6
When this bit is set to one all characters entered on the paddles will be echoed back to the host. From the host's perspective, paddle echo and serial echo are the same; in both cases the letter sent in Morse by Winkeyer2 is echoed back to the host. The echo occurs after the letter has been completely sent. The host can determine the source by the sense of the “break-in” status bit. If the bit is high when the echoed letter comes in then the letter’s source was from the paddles, if break-in is low the source if from the serial port.

Bit 5,4
Winkeyer2 supports Iambic A, B, Ultimatic, and Bug keying modes. In iambic mode Winkeyer2 makes both dits and dahs automatically based on which paddle you press. In bug mode Winkeyer2 makes the dits and you make the dahs. You also can use bug mode to operate in straight key mode or if you want to key through Winkeyer2 with a different keyer, simply set bug mode and use the dah input to key Winkeyer2.

In either iambic mode, alternating dits and dahs are sent while both paddles are held closed. In mode B an extra alternate dit or dah is sent after both paddles are released. In Ultimatic mode when both paddles are pressed the keyer will send a continuous stream of whichever paddle was last pressed.

Bit 3
Paddle swap: this is a nice feature to have when right and left handed ops want to share the same keyer.

Bit 2
Echo back is a feature that is included to allow a host application to stay exactly in sync with Morse letters sent. When this mode is enabled all data taken out of the serial buffer is sent to the host after it has been sent in Morse. This allows the host to reconcile differences in timing introduced by Winkeyer2's internal 32 byte serial buffer. Note that only letters, and not buffered commands with their parameters or wordspaces, are echoed back to the host.

Bit 1
Here is how autospace works: If you pause for more than one dit time between a dit or dah Winkeyer2 will interpret this as a letter-space and will not send the next dit or dah until full letter-space time has been met. The normal letter-space is 3 dit spaces. Winkeyer2 has a paddle event memory so that you can enter dits or dahs during the inter-letter space and Winkeyer2 will send them as they were entered. With a little practice, autospace will help you to send near perfect Morse.

Bit 0
Winkeyer2 supports contest spacing which reduces the wordspace time by one dit. Instead of 7 dits per wordspace, Contest spacing selects six dits per wordspace.

● Load Defaults <0F><value list> value list is a set of 15 binary values

This command is provided to allow all the operating parameters to be loaded into Winkeyer2 in one block transfer. The values are binary and must be loaded in order. The values are exactly the same as those loaded for the individual commands. The preferred time to issue this command is at reset just after the interface has been opened. Issuing this command while sending Morse is not advised.

1) Mode Register	2) Speed in WPM	3) Sidetone Frequency
4) Weight	5) Lead-In Time	6) Tail Time
7) MinWPM	8) WPM Range	9) 1st Extension
10) Key Compensation	11) Farnsworth WPM	12) Paddle Setpoint
13) Dit/Dah Ratio	14) Pin Configuration	15) Don't care, set to 0

Table 9 - Default Value List in order of issuance

● Set 1st Extension <10><nn> nn is in the range of (0 to 250) $\times$ 1 mSecs

Example: <04><80> sets lead in to 80 mSecs

Winkeyer2 addresses a problem often encountered when keying older transceivers that have a slow break-in response. Due to a slow receive to transmit changeover time, the first dit or dah of a letter sequence can be chopped and reduced in length. Adding a fixed amount to the first element of a sequence can compensate for this. For example, an R would be sent with the first dit elongated but the subsequent dah-dit sent normally. The compensation amount is transceiver dependent and is generally independent of sending speed. Note though that this is usually only a noticeable problem at higher CW speeds >25 WPM.

A challenge in this scheme is to determine when sending has stopped long enough to cause the transceiver to switch back to receive. If it has it'll require a new first element correction on the next sequence. Winkeyer2 uses the PTT tail timer to determine this, set the tail timer to roughly match the transmit to receive changeover time of the transceiver and things will work fine. It takes some trial and error to get it set up right so make sure you preserve the value and load it as a defaults after reset.

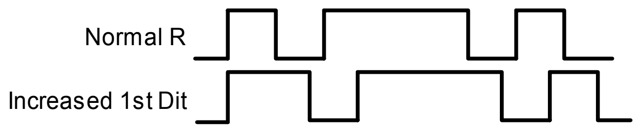


Figure 7 – 1st Extension Example

● Set Key Comp <11><nn> nn is in the range of (0 to 250) × 1 mSecs

Example: <11><B4> sets key comp to 180 mSecs

Keying Compensation allows a fixed amount to be added to the length of all dits and dahs. QSK keying on modern transceivers can cause shortening of the dit and dah elements which is especially noticeable at high speeds. Winkeyer2 allows the length of the dit and dah elements to be increased uniformly to compensate for this. The adjustments are made in units of one-millisecond steps. The maximum adjustment is 250 mSecs. Key compensation is very similar to Weighting in that any adjustment added to the dits and dahs is subtracted from the spacing so the speed is not changed. The difference between weighting and compensation is that compensation is independent of speed, so if 10 msec of key compensation is selected 10 msec will be always be added regardless of speed. So be careful at high speeds and large values of key compensation, you may end up with no inter-element space.

When nn = 00 there is no adjustment while nn=12 will add twelve mSecs.

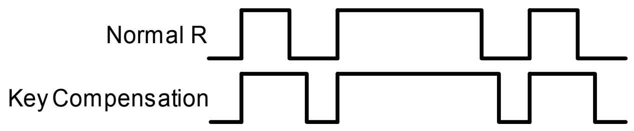


Figure 8 – Keying Compensation Example

● Set Paddle Switchpoint <12><nn> nn is in the range of 10-90%

Example: <03><37> for sensitivity=55

This controls when Winkeyer2 will start looking for a new paddle press after sensing the current one. If there is not enough delay the keyer will send unwanted dits or dahs, if there is too much delay it bogs you down because you can't get ahead of the keyer. The default value is one dit time (50) and is adjustable in percent of a dit time. Faster operators report a setting somewhat less than default is more pleasing. If the paddle sensitivity is set to zero, dit and dah paddle memory is disabled. The delay is calculated with this formula:

DELAY_TIME = (SWITCHPOINT×DIT_TIME)/50 where Switchpoint is a value between 10 and 90.

● Null Command <13> This command is a NOP

● Software Paddle <14><nn> nn = 00 paddle up, n=01 dit, n=02 dah, n=03 both

This command provides a way to assert paddle inputs from the host. The PC application would convert key down codes to Software Paddle commands. Due to the limited response time of the keyboard, operating system, and serial communication the best you can do is around 20 WPM. The paddle watchdog will be used for this interface as if it were a normal paddle input.

● Request Winkeyer2 Status <15> Return Winkeyer2's status byte

This command is used to queue a request to Winkeyer2 to send its current operating state. The status byte returned consists of a bit field that is defined by the following table. The three MSBs of the status byte are always 110. Note that in WK2 mode bit 3 identifies the status byte as a pushbutton status byte. WK2 mode is set by the ADMIN command 11 before WK is opened.

Status Bit	Name	Definition
7 (MSB)	Tag	1
6	Tag	1
5	Tag	0
4	WAIT	WK is waiting for an internally timed event to finish
3	KEYDOWN	Key down status (Tune) 1 = key down
2	BUSY	Keyer is busy sending Morse when = 1
1	BREAKIN	Paddle break-in active when = 1
0 (LSB)	XOFF	Buffer is more than 2/3 full when = 1

Table 10 – Winkeyer2 Status Definition WK1 compatible mode

Status Bit	Name	Definition
7 (MSB)	Tag	1
6	Tag	1
5	Tag	0
4	WAIT	WK is waiting for an internally timed event to finish
3	0	This is a WK Status Byte
2	BUSY	Keyer is busy sending Morse when = 1
1	BREAKIN	Paddle break-in active when = 1
0 (LSB)	XOFF	Buffer is more than 2/3 full when = 1

Table 11 – Winkeyer2 Status Definitions (WK2 Mode)

Status Bit	Name	Definition
7 (MSB)	Tag	1
6	Tag	1
5	Tag	0
4	PB4STAT	1 when PB4 pressed, 0 when PB4 unpressed
3	1	This is a pushbutton status byte
2	PB3STAT	1 when PB3 pressed, 0 when PB3 unpressed
1	PB2STAT	1 when PB2 pressed, 0 when PB2 unpressed
0 (LSB)	PB1STAT	1 when PB1 pressed, 0 when PB1 unpressed

Table 12 – Winkeyer2 PB Status Definitions (WK2 Mode)

The PB status byte will be sent whenever the PB state changes, in other words a byte will be sent when a pushbutton is pressed and a byte will be sent when the pushbutton is released. Only one press will be detected at a time.

● Pointer Cmd <16><nn> Input Buffer Command Set

This command allows the host app to manipulate the input buffer for special situations such as "on the fly" callsign correction. Four commands make up the pointer command set:

nn=00	Reset input buffer pointers to start of buffer, only issue this when buffer is empty.
nn=01	Move input pointer to new position in overwrite mode
nn=02	Move input pointer to new position in append mode
nn=03	Add multiple nulls to the buffer <16><03><number of nulls>

A detailed description of the pointer command is detailed in a separate application note.

● Set Dit/Dah Ratio <17><nn> nn is in the range of 33-66

Example: <03><42> for ratio=1:4

Modifies the ratio of dit time to dah time, standard is 1:3 (dit:dah). The formula to determine dah/dit ratio is:

$$DAH/DIT = 3*(nn/50)$$

A value of 50 selects 1:3, a value of 33 would select 1:2, and a value of 66 would select 1:4. This causes an intentional distortion of the Morse waveform. Some ops use this option to make their CW sound less "machine like". *A little goes a long way!!*

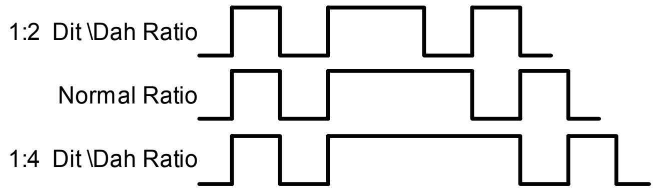


Figure 9 – Three ratio settings for the letter R

Buffered Commands

These commands go into the input buffer maintaining their positional relationship to data.

● PTT On/Off <18><nn> nn = 01 PTT on, n = 00 PTT off

This command allows the PTT output to be used for a custom purpose. The command is operational only when sidetone and PTT are disabled (See PINCFG command) PTT can be turned on or off at will and will be unaffected by all other commands including Clear Buffer. Typical applications could be as a power level control, antenna selector, or to turn on a cooling fan. Since this is a buffered command, the on/off will happen at the command's position in the buffer and remain in effect until the next PTT ON/OFF command is encountered. This command will not stall the output buffer.

● Key Buffered <19><nn> nn = 0 to 99 seconds

Use this command to assert the key output for a specific period of time. Since this is a buffered command, the key down will begin at the command's position in the buffer and will stall the buffer until the timeout has been satisfied. The key down can be aborted either by the paddles or by a Clear Buffer command. The maximum allowable key down time is 99 seconds.

● Wait for nn Seconds <1A><nn> nn = 0 to 99 seconds

This command is used to insert a fixed pause into a message. Since this is a buffered command, the pause will begin at the command's position in the buffer and will stall the buffer until the timeout has been satisfied.

● Merge Letters <1B>[C][C] Merge Two Letters into a Prosign

You can build "on the fly" prosigns with this command. Issue the merge command followed by two letters or numbers and they will be merged together: <1B> [A] [R] is sent as **AR.** Note that nothing will be sent until both letters have been received. Several common prosigns such as AR, SK, BT, and DN are already assigned (see page 16) One application of this feature is to send special European language characters.

● Buffered Speed Change <1C><nn> nn is in the range of 5-99 WPM

Example: <02><23> set 35 WPM

This command places a speed change command into the serial buffer that will be acted upon when it is taken out of the buffer. The current speed in force will be stored and will be reinstated when the buffered speed change is cancelled by a Cancel Speed Change command or any of the following: Unbuffered Speed change, Weight change, Farnsworth change, Ratio change, Compensation change, or Mode change.

This command is useful for building messages with embedded speed changes. In this example the first part of the message will be sent at 5 WPM, the second at 25 WPM and the end at whatever the current speed is:

```
<1C><05>VVV DE K1EL <1C><19> VVV DE K1EL<1E><END DE K1EL>
```

● HSCW Speed Change <1D><nn> nn = (lpm/100)

This command acts the same as the immediate HSCW command. This allows you to insert an HSCW burst in a regular CW message or to put HSCW bursts of two different rates into the same message.

● Cancel Buffered Speed Change <1E> No Parameter

This command will cancel any buffered speed change command that is in force. The sending speed that was in force before any buffered speed change was encountered will be restored. Several buffered speed changes can be issued within a message but none will alter the original sending speed.

● Buffered NOP <1F> No Parameter

This command will occupy a position in the input buffer but will be ignored when it is processed.

PTT Functionality

WKUSB's PTT output is used to control an accessory device in addition to normal CW transmitter keying. In most cases this device is a linear amplifier but it could be an antenna changeover relay or PTT input of the transmitter. In all of these cases there are delay requirements that must be met to insure that the accessory device is switched on before transmission begins and is held on until transmission completes. This prevents damage to the accessory device due to hot switching. WK2 provides three independent PTT delays to meet this requirement. (See figures on Page7)

The first is Lead-In delay. This starts a keying event. PTT will be asserted first and then, after the Lead-In delay expires, the key output will be asserted. Lead-In can be set to a value from 0 to 250 milliseconds in 10 millisecond steps (0 to 250).

Tail Delay specifies the amount of time PTT will be released after Key is released. Like Lead-In delay, it is adjustable from zero to 250 milliseconds in 10 millisecond steps. Tail delay is the sum of two delays, Tail setting times 10 milliseconds plus three dit times plus.

In setting tail delay there are two considerations, the first is to set the delay to prevent hot switching and the second is to optionally add to the delay to fill in between letters sent by paddle. Most ops don't want their amplifier to be switched in and out between letters. An issue arises in that Tail delay is not solely proportional to sending speed. This is problematic when someone sets a comfortable tail delay at a slow speed and then increases the speed to a much faster rate. At the faster rate, the tail delay will hold too long after keying stops. Alternatively, if a comfortable delay is set at a faster WPM rate, PTT will then drop out between letters at a slower speed. Since it is very time consuming to constantly adjust the tail delay with sending speed, a different delay method is used for paddle sending, namely Hang Time.

Hang Time is adjustable in four steps and is measured only in wordspace and dit times rather than fixed milliseconds. This means that the delay will track sending speed. In other words it will decrease automatically as sending speed increases and vice versa. The four settings for Hang Time are:

HangTime = 0: wait 1 wordspace + 1 dit before ending paddle insertion
HangTime = 1: wait 1 wordspace + 2 dits before ending paddle insertion
HangTime = 2: wait 1 wordspace + 4 dits before ending paddle insertion
HangTime = 3: wait 1 wordspace + 8 dits before ending paddle insertion

A major advantage to this scheme is that Tail delay can set to provide just enough delay to prevent hot switching for machine sent CW while hang delay is designed to hold PTT between letters and in general is much longer than tail delay. This means that PTT will switch off faster for machine sent CW. Note that PTT will automatically be held between machine sent letters and words but will drop out quickly after a message has been sent as long as there is no trailing wordspace.

Unsolicited Status Transmission

Winkeyer2 will send two types of unsolicited status to the host: speed pot change, and WK status byte change. Whenever the speed pot is moved, its new value will be sent to the host. Likewise whenever there is a change to the internal status register inside Winkeyer2, a copy of it will be sent to the host. In WK2 mode, the WK status byte is expanded to include pushbutton status.

The status byte will be in the same format as previously described in the Get Status command. Likewise the speed pot status will be as described in the Get Pot command. Since these bytes can arrive at any time and potentially can be mixed with echo back bytes, they have identifying tags. If the MSB is set that identifies the byte as unsolicited, bit 6 then identifies either a speed pot byte or a status byte.

The host can force either of these bytes to be returned by using their respective Get Pot or Get Status commands. Due to the parallel task handling nature of Winkeyer2 a response may not be immediate, there may be another byte in the return queue that needs to be sent to the host first. Worst case latency will be 200 milliseconds. It is not advisable for the host to wait for a response, it is better to handle it as illustrated by the code fragment presented on page three.

Prosign Key Assignments

Winkeyer2 has mapped several unused character codes to standard prosigns. Table 5 shows the mappings. Any additional prosigns can easily be generated using the merge character command.

ASCII	Hex		Prosign		ASCII	Hex		Prosign
“	0x22	*Is mapped to*	RR		+	0x2B	*Is mapped to*	AR
#	0x23	*Is mapped to*	EE (null)		-	0x2D	*Is mapped to*	DU
$	0x24	*Is mapped to*	SX		/	0x2F	*Is mapped to*	DN
%	0x25	*Is mapped to*	EE (null)		:	0x3A	*Is mapped to*	KN
&	0x26	*Is mapped to*	EE (null)		;	0x3B	*Is mapped to*	AA
‘	0x27	*Is mapped to*	WG		<	0x3C	*Is mapped to*	AR
(	0x28	*Is mapped to*	KN		=	0x3D	*Is mapped to*	BT
)	0x29	*Is mapped to*	KK		>	0x3E	*Is mapped to*	SK
*	0x2A	*Is mapped to*	EE (null)		@	0x40	*Is mapped to*	AC

Table 13 – Prosign/Abbreviations Assignments

Serial Baud Rate

Winkeyer2’s standard baud rate is 1200 baud with an alternate baud rate of 9600 selected by command. The communication settings are eight data bits, 2 stop bits, with no parity.

Gap (Extra Space) Insertion

Winkeyer2 interprets the | character (hex 0x7C) as a ½ dit delay time. The | character can be included in a text string to add extra emphasis to similar sounding sequences. An example is W1OMO, sending it as W1|O|M|O makes it easier to copy.

Reset WK2/Restore Factory Defaults

As mentioned previously, the command pushbutton can restore Winkeyer2 operation if a lockup has occurred. Lockup can happen if Winkeyer2 is disconnected from the host unexpectedly while the host is actively communicating to Winkeyer2. It's very unlikely a lockup will occur but it can happen so a means is provided to recover.

Normally, you press the command button and wait for an **R** before entering a command. If you continue to press the command button after the **R** is sent, in about five seconds WK2 will send 6 dits in sidetone and then reboot back to standalone mode and reload the standalone settings. You can then replug it back into the PC and reopen it from you app or just use it in standalone mode. Note that the standalone message contents are not erased but all other settings will be restored back to the last standalone settings you had preserved using the **P** command.

If you have a case where you want to return WK2 back to factory defaults, a special command sequence is provided. Press the command button, wait for the **R** and then enter di-dah-di-dah. When WK2 responds with an **R,** re-enter di-dah-di-dah and WK2 will clear all settings including messages and return to factory defaults. After the initialization is complete WK2 will send dah-dah-dah-dit. Note that this will also erase stored messages.

Sleep Mode

WK2 utilizes the low power sleep mode of the PIC CPU. WK2 normally rests in sleep mode and draws about 1 μA of DC current. When either of the paddles or a push-button is pressed, the chip wakes up and goes into active mode drawing less than 1 ma idle and about 10ma while actively sending and driving sidetone. After the paddle or push-button is serviced WK2 goes back to sleep after a few seconds. Note that running WKUSB without sidetone and instead utilizing your radio's sidetone will extend battery life.

Keyer Lock

A lock feature is provided to disable paddle input and message pushbuttons. This is useful when you want to pack up the keyer and effectively turn it off. Other times it's nice to lock the keyer paddles to keep little hands from sending "messages". While the keyer is locked it is held in low power shutdown mode. To lock the keyer press the command pushbutton, wait for the R, and then enter a period (di-dah-di-dah-di-dah). To unlock the keyer press and hold the command pushbutton for about 5 seconds and WK2 will wake up and send an R.

WK2 Host Mode Command Table

Command Name	Code	Type	Description	Syntax	Pg
Admin	00	Imm	Administative Commands	<00><type>…	5
Sidetone Freq	01	Imm	Set sidetone frequency	<01><freq>	6
Speed	02	Imm	Set Morse sending speed	<02><WPM>	6
Weighting	03	Imm	Set key weighting	<03><weight>	6
PTT Lead-in/Tail	04	Imm	Set up PTT delays	<04><leadin><tail>	7
Speed Pot Setup	05	Imm	Set up speed pot range	<05><m><wr><pr>	7
Pause	06	Imm	Pause Morse output	<06><0 or 1>	7
Get Speed Pot	07	Imm	Request speed pot value	<07>	7
Backspace	08	Imm	Backup input pointer	<08>	8
Pin Configuration	09	Imm	Set output pin configuration	<09><config>	8
Clear Buffer	0A	Imm	Clear input buffer	<0A>	9
Key Immediate	0B	Imm	Direct control of key output	<0B><0 or 1>	9
HSCW Speed	0C	Imm	Set HSCW speed	<0C><lpm/100>	9
Farnsworth	0D	Imm	Set Farnsworth speed	<0D><WPM>	9
Winkeyer2 Mode	0E	Imm	Load Winkeyer2 mode byte	<0E><mode>	9
Load Defaults	0F	Imm	Download WK state block	<0F><…15 values…>	10
First Extension	10	Imm	Setup 1st element correction	<10><msec>	10
Key Compensation	11	Imm	Set Keying Compensation	<11><comp>	11
Paddle Switchpoint	12	Imm	Setup paddle sensitivity	<12><sens>	11
Null	13	Imm	Null Command, NOP	<13>	11
S/W Paddle Input	14	Imm	Software Paddle Control	<14><paddle select>	11
Winkeyer2 Status	15	Imm	Request Winkeyer2 status	<15>	12
Buffer Pointer	16	Imm	Buffer pointer commands	<16><cmd>…	12
Dit/Dah Ratio	17	Imm	Set ratio of dit/dah	<17><ratio>	13
PTT Control	18	Buff	Turn PTT on/off	<18><0 or 1>	13
Timed Key Down	19	Buff	Turn KeyOut on for an interval	<19><secs>	13
Wait	1A	Buff	Wait for N seconds	<1A><secs>	13
Merge Letters	1B	Buff	Merge chars into prosign	<1B>[c][c]	13
Speed Change	1C	Buff	Change Morse speed	<1C><WPM>	14
HSCW Speed	1D	Buff	Set HSCW speed	<1D><lpm/100>	14
Port Select	1D	Buff	Select output port	<1D><n> n=0 or n=1	14
Cancel Buff Speed	1E	Buff	Cancel Buff Speed Change	<1E>	14
Buffered NOP	1F	Buff	Null Command (buffered)	<1F>	14

Table 14 – WK Host Mode Command Table

Winkeyer2 Standalone Mode

FEATURES

- Keyer speed range: 5 - 99 WPM
- HSCW: 1000, 1500, 2000, 3000, 4000 or 6000 lpm
- QRSS: 3, 6, 10, 12, 30, 60 second dits
- Non-Volatile Message Memory: 232 letters in six slots with embedded commands.
- Dynamically allocated message memory
- Backspace supported on message entry
- Keying Modes: Bug, Ultimatic, Iambic A or B
- Serial Number Generation
- Audio Frequency keying mode (PTT)
- Adjustable Weight 25 to 75 %
- Automatic letterspace mode (Autospace)
- Message Stacking
- Adjustable letterspace
- Adjustable Keying Compensation 0 to 31 mSec
- Paddle swap command
- Beacon: Programmable interval: 1 to 99 seconds
- Sidetone Output: TTL Square wave, 100Ω output Z
- Adjustable Sidetone frequency
- Keying/PTT outputs: TTL, high true when keyed
- Speed control potentiometer support
- Push-button user interface
- 22 easy to use commands
- Operating Voltage: 2.5-5.5 VDC, built in oscillator
- Power Consumption: <1 ma active, 1 μA standby
- Downloadable messages

When not connected to a host WK2 will operate in standalone mode. When in standalone mode WK2 closely emulates the K12 keyer IC in functionality. A few enhancements have been added which will be described. The most noticeable difference between host and standalone mode is that when in standalone mode the user can enter commands on the paddles. This is initiated by pressing the command pushbutton. In host mode the pushbuttons are ignored. Low power mode is activated in standalone mode, this means that WK2 will go into a low power sleep state when idle. This makes it very battery friendly.

Standalone Pushbutton Functionality

Winkeyer2 standalone requires at least one push-button control, this switch is referred to as the command push-button and is connected to pin 13. It serves two functions, command control and message record/playback control. Up to five additional message push-buttons can be added to provide a total of six message slots. Be sure to use normally open switches for the push-buttons. Pin 13 is an analog input which senses the switch network shown in Fig 22. Message push-buttons 2 through 6 are connected as shown. Use 5% tolerance resistors for the switching network.

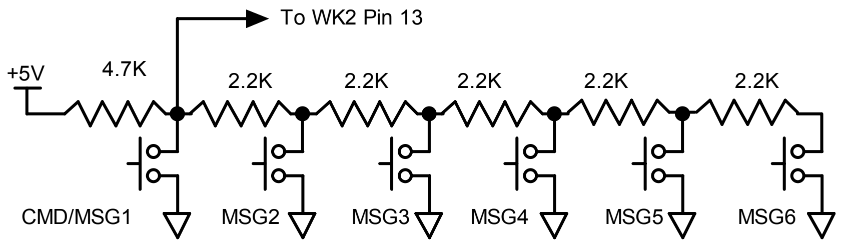


Figure 10 – Pushbutton Matrix

Standalone Command Mode

If the command push-button is pressed and held, the WK2 will respond after about two seconds with the letter ***R*** in sidetone only. This means WK2 is ready to accept a command, you simply enter the command letter in Morse on the paddles and the command will be executed. Some commands require additional information which WK2 will prompt you for by outputting the letter ***E*** (for enter). All commands provide some sort of feedback to tell you if the command was understood and executed properly, in most cases an ***R***. If an illegal command is entered WK2 will respond with a question mark.

Important Note ! When in command mode, transmitter keying is disabled and replies are sent in sidetone only. Thus in order to use command mode you must have a sidetone speaker of some sort. If sidetone had been disabled with the **A** command it will be re-enabled automatically when entering command mode.

WK2 Standalone Command List

A -	**Select sidetone on or off**	**O -**	**Select output key port**
B -	**Set speed control bottom WPM**	**P -**	**Preserve Settings**
C -	**Set command speed in WPM**	**Q -**	**Query current settings**
D -	**Decrement serial number**	**R -**	**Review message without transmitting**
F -	**Set Farnsworth Speed**	**S -**	**Set speed control range in WPM**
G -	**Select serial number 0/9 format**	**T -**	**Load PTT Tail Time**
H -	**Set Paddle Hang Time**	**U -**	**Select Autospacing on/off**
I -	**Set Letterspace Adjust**	**V -**	**Set Keying compensation in mSec**
J -	**Set Paddle sensitivity**	**W -**	**Set Key Weight**
K -	**Select keyer mode**	**X -**	**Exchange Paddles**
L -	**Set PTT Lead-In Time**	**Y -**	**Set Dit/Dah Ratio**
M -	**Mute Transmit (CPO mode)**	**Z -**	**Select sidetone frequency**
N -	**Load 4 digit serial number**		

In the command descriptions below, the **[n]** or **[nn]** notation means that additional parameters must be entered on the paddles after the command. A letter displayed in **BOLD** is something you enter, ***BOLD ITALIC*** is what WK2 responds with. A [pb] means that WK2 will wait for you to press one of the message pushbuttons.

A - Sidetone enable is toggled when this command is entered. Toggle means if the sidetone was on when this command was issued it will be turned off and vice versa. WK2 will acknowledge this command by responding with an ***R***. Note: If sidetone is disabled it will be re-enabled while in command mode.

B [nn] - Speed Pot Bottom is used to set the bottom of the speed control range, when the speed control is turned fully counter clockwise. The default setting is 5 WPM but you can move the bottom WPM value to something that is more comfortable. For example if your primary operation speed range starts at 12 WPM you can set the speed bottom to 12 WPM and that will give you a range from 12 to 42 WPM, with the Range value set at the default of 30 WPM. The range value can be altered using the **S** command. It doesn't matter where the speed pot is set when you enter either command, but odds are the current operating speed will be affected.

The bottom or range values are entered directly in WPM in the following manner. For example, after a **B** command is issued WK2 will respond with a single ***E***, you then directly enter the speed. As a short cut, a ***T*** can be entered for zero. If the desired speed is a single digit, either enter the single number, or send a zero or T first. i.e. **7** or **07** or **T7** will all give you 7 WPM. Likewise, **2T** can be entered for 20 WPM. If an illegal value is entered, WK2 will respond with a question mark, if the value is good WK2 will respond with an ***R***.

More details can be found in the Speed Pot Functionality section on Page 22 as well as the **S** command below.

C [nn] - Command Speed sets the command WPM speed of WK2. WK2 can use different speeds for command and transmit Morse speeds. Changes in transmit speed will not affect command speed. After the **C** command is issued enter the speed in WPM. *See the* **B** *command for details on entering Morse speeds.*

D - Decrement Serial Number by 1 WK2 responds with an ***E*** after the decrement.

F [nn] - Farnsworth WPM spacing is useful for CW practice because it encourages you to learn characters by sound not individual dits and dahs. In WK2, Farnsworth is implemented by sending letters at a fixed rate of **nn** WPM regardless what the WPM sending rate is. Spacing between characters is determined by the sending rate. When the WPM rate is set above the Farnsworth WPM, Farnsworth is disabled.

G - Toggle Serial Number Format for 0 and 9 WK2 responds ***N*** for normal, ***A*** for 0 sent as T and 9 sent as N

H [n] - Set Transmit PTT Hang Delay Time. You can select one of four delays:

HangTime = 0: wait 1 wordspace + 1 dit before ending paddle insertion
HangTime = 1: wait 1 wordspace + 2 dits before ending paddle insertion
HangTime = 2: wait 1 wordspace + 4 dits before ending paddle insertion
HangTime = 3: wait 1 wordspace + 8 dits before ending paddle insertion

After entering the command letter, you will be prompted with an **E** to enter the desired hang time as a number 0 to 3 as indicated in the table.

I [nn] - Set Letterspace Adjustment where **n** is a value, 0 to 15, specifying an additional letterspace to be applied between letters. Multiply **n** by two to arrive at the actual adjustment percentage. For example a value of 7 applies 14% additional letterspace between letters. The maximum adjustment is 30%.

J [nn] - Paddle Sample Point controls when WK2 will start looking for a new paddle press after sensing the current one. If there is not enough delay the keyer will send unwanted dits or dahs, if there is too much delay it may slow you down because you can't send ahead of the keyer. The default value is one dit time (50) and is adjustable in a fraction of a dit time. Faster operators report a setting somewhat less than default is more pleasing. **If the paddle sensitivity is set to zero, both dit and dah paddle memories are disabled.** The delay is calculated with this formula:

DELAY_TIME = (nn×DIT_TIME)/50 where switch point is a value between 01 and 99.

K - Set Keying Mode There are six different keying modes supported by WK2: Iambic mode A and B, Straight Key, Bug, Ultimatic, Dit priority mode, and Dah priority mode. In either iambic mode, alternating dits and dahs are sent while both paddles are held closed. In mode B an extra alternate dit or dah is sent after both paddles are released. In straight key mode a dah paddle press will key the transmitter for as long as the paddle is pressed. Use the swap command: **X** to choose either the left or right paddle Bug mode directly keys with the dah paddle and generates dits automatically when the dit paddle is pressed. In Ultimatic mode when both paddles are pressed the keyer will send a continuous stream of whichever paddle was last pressed. Dit and dah priority mode will generate dits and dahs automatically in response to single paddle presses, but when both paddles are pressed either dit or dah has priority.

After the **K** command is issued the current mode is set by entering a single letter:

Iambic B:	Enter B
Iambic A:	Enter A
Ultimatic:	Enter U
Straight Key:	Enter S
Dit Priority:	Enter E
Dah Priority:	Enter T

L [nn] - PTT Lead In Time can be set to a value between 0 and 99 which is a subset of the entire possible range of 0 to 255. See the Set PTT Lead/Tail description on page 7 for more information.

M - Toggle Transmit Mute is useful for off line practice. WK2 responds with ***R*** for mute on and N when mute off.

N [nnnn] - Load 4 Digit Serial Number All four digits must be entered including leading zeroes. The serial number is played by inserting a play message token /N into a message. The serial number is automatically incremented after playing. See *Embedded Command* section for more details.

O - Toggle Key Output Port Each time the **O** command is issued the key port is toggled back and forth between 1 and 2. When port 1 is selected a single dit is echoed and two dits are echoed for port 2.

P - Save Current Settings in EEPROM After the values have been stored WK2 will respond with an R. Note that messages are always stored in EEPROM when entered and do not require a P command to save them. When changing from host mode back to standalone mode the last settings stored in EEPROM will be restored.

Q - Query WK2 Current Settings After the command is issued WK2 will respond with WK2 commands and their settings sent in the following format:

WPM	*is sent first*
N	*followed by Serial Number*
M	*followed by free msg memory space in letters available*
C	*followed by command WPM*
W	*followed by weight*
L	*followed by lead time*
T	*followed by tail time*
X	*followed by 1st extension (this parameter described in host mode section)*
V	*followed by key compensation*
F	*followed by Farnsworth WPM*
J	*followed by Paddle Sample Adjust*
Y	*followed by dit/dah ratio*

S	*followed by speed pot min WPM*
G	*followed by pot range (this parameter described in host mode section)*

You can abort this command at any time after the first parameter is sent by pressing the Command and PB4 pushbuttons together or holding either the left or right paddle.

R [pb] - Review a Message Without Transmitting After the **R** command is entered WK2 will respond with an ***M***. Then press the message button of the message you wish to play. The message will be sent in sidetone only. If you try to play an empty slot WK2 will respond with ***MT***. Embedded commands will be sent as is without expansion. Don't forget that sidetone has to be enabled to use this command.

S [nn] - Set Speed Control Sweep Range The default setting is 30 WPM, meaning that the speed control will cover a range of 30 WPM from one end to the other. The minimum WPM is set by the speed bottom command **B**, with the default being 5 WPM. Thus the default speed range is 5 to 35 WPM. The minimum range setting is 5 WPM. See the **B** command for more information on entering numeric WPM values.

T [nn] - Set PTT Tail time Value can be set to a value between 0 and 99 which is a subset of the entire possible range of 0 to 255. See the Set PTT Lead/Tail description on page 7 for more information.

U - Turn Autospace Mode Off and On When autospace is enabled WK2 will automatically insert proper inter-letter space between letters. Each time the **U** command is issued WK2 will toggle between modes responding with an ***A*** for autospace enabled an ***N*** for autospace disabled.

Here is how autospace works: If you pause for more than one dit time between a dit or dah WK2 will interpret this as a letter-space and will not send the next dit or dah until the letter-space time has been met. The normal letter-space is 3 dit spaces but this can be increased by using the **I** command. WK2 has a paddle event memory so that you can enter dits or dahs during the inter-letter space and WK2 will send them as they were entered. With a little practice, autospace will help you to send near perfect Morse.

V [nn] - Keying Compensation allows a fixed amount of time to be added to the length of all dits and dahs. QSK keying on modern transceivers can cause shortening of these elements which is especially noticeable at high speeds. WK2 allows the length of the elements to be increased uniformly to compensate for this. The adjustments can be made in one-millisecond steps. The maximum adjustment is 31 mSecs. Key compensation is very similar to Weighting in that any adjustment added to the dits and dahs is subtracted from the spacing so the resulting speed is not changed. The difference between weighting and keying compensation is that compensation is independent of speed, so if 10 mSec of key compensation is selected, 10 mSec will be always be added regardless of speed. So be careful at high speeds with large values of keying compensation, dits and dahs may run together with no spacing at all.

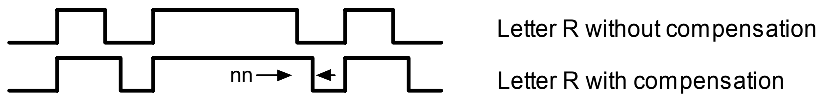


Figure 11 – Key Compensation

W [nn] - Keying Weight can be adjusted in percentage from 25% to 75%. When set to 50 % the dit time is equal to the inter-element time, which is normal. Values less than 50 reduce weighting while values greater than 50 increase weighting. Note that weighting does not affect sending speed because any increase in keyed time is subtracted from spacing time. Reduction in weighting results in a thinner sound while increased weighting results in a heavier sound. Since weighting tracks speed, a given weighting will sound the same at all speeds.

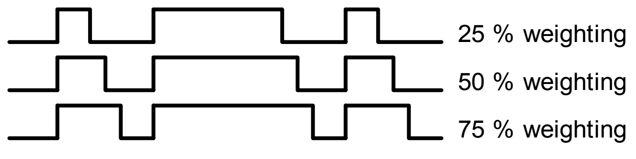


Figure 12 – Key Weighting

X - Exchange Paddle Inputs (dit and dah) WK2 will respond with a letter ***R*** to signify that this command was accepted.

Y [nn] - Set Dit/Dah Ratio nn is in the range of 33-66. Entering **Y50** sets the standard 1:3 ratio. The full range is **Y33** for 1:2 up to **Y66** for 1:4 dit/dah ratio.

Z - Change Sidetone Frequency After this command is entered the sidetone oscillator will be turned on. Pressing the paddles will raise or lower the frequency. There are 10 possible choices: 2000, 1333, 1000, 800, 666, 570, 500, 440, and 400 Hz. Pressing the command push-button will end this command and store the new sidetone frequency. Due to the multi-threaded processing architecture of WK2, continuous sidetone frequency selection is not possible.

Standalone Speed Potentiometer Functionality

WK2 uses a 10K potentiometer connected to pin 10 to set sending speed. Turning the speed control will change the speed and update the WPM rate immediately with minimal lag. The entire sweep of the speed pot is called the speed window and it can be modified with two commands. The **B** command sets the bottom, or lowest point of the sweep. The **S** command sets the WPM range. Generally an operator sets the speed bottom first to set the starting point and then enters a range value which determines the upper WPM. Entering 10 WPM for the bottom with a range of 40 WPM will specify a window that starts at 10 WPM and sweeps up to 50 WPM. This allows you to tailor the speed control to an area that you prefer, but it can be changed at any time. The minimum acceptable values for bottom and range are 5 WPM. A 5 WPM range is rather useless, better ranges are in the vicinity of 20 to 40 WPM. If it is desired to set the speed to an exact speed, the easiest way to do this is to turn the speed control fully counter clockwise and set the speed bottom to the desired speed. Otherwise you can determine the current speed control setting with the **Q** (query) command.

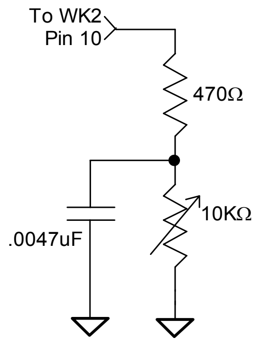


Figure 13

Standalone Message Functionality

Messages are loaded by pressing the command button until a WK2 responds with an ***R***, then press the message pushbutton of the memory slot you wish to enter. When WK2 is ready to accept a new message it will respond with an ***E***. If you wait too long WK2 will respond with a ***?*** and you have to start over. In some keyers, such as WKUSB, only four pushbuttons are provided, messages 5 and 6 are accessed by pressing two pushbuttons in the following sequence after the ***R*** is echoed:

Press and hold either message button 2 or 3. (for message 5 or 6 respectively)
Press the command pushbutton.
Release both push buttons.

(Message 5 and 6 can be played in a similar manner by first pressing and holding pushbutton 2 or 3, then pressing the command pushbutton, and finally releasing both)

After WK2 responds with an ***E***, a message is entered directly on the paddles at a steady rate, making sure to leave proper space between letters. To insert a word space simply pause for longer than a word space and WK2 will respond with an ***E*** to signify a word space insertion. You can force a wordspace insertion by entering di-di-dah-dah. This allows you to put a wordspace at the beginning of a message or insert more than one wordspace in a row. A ½ letterspace pad character can be inserted by entering di-di-dah-dah-dit.

When the message has been completely entered, press the command push-button or enter di-dah-di-dah and WK2 will respond with an ***R*** to signify that the message was accepted and stored. If a mistake is made while entering a message, press and hold the command pushbutton and WK2 will backspace through the letters that have been entered. When you reach the position you want, release the button and new letters can then be added after that position. If the message memory becomes full while entering a message, WK2 will stop further loading, respond with an ***F***, and return WK2 back to non-command mode. There are 232 letters in message memory that can be distributed in any way between six message slots. The length of the individual message slots is not fixed. This means, for example, you could have one message of 80 characters, one message with 5 characters, and a third with 10 characters and still have 141 locations left to split among the remaining three slots. Keep in mind that each word space occupies one memory location.

Note that usually when you stop at the end of a message WK2 will enter a wordspace before you have a chance to press the command pushbutton. There are cases when you do not want a wordspace at the end, especially if you are using PTT to key an amplifier. This is because WK2 will hold PTT during that added wordspace which will delay an exchange turnaround. The best way around this is to end a message with the di-dah-di-dah character inserted immediately after the last letter in the message. This ends message entry mode immediately.

If you are having problems loading messages into WK2, make sure you leave adequate space between letters and are not sending much faster or slower than current command speed. If, for example, you enter an ***A*** followed by a ***T*** and end up with a ***W***, you are not allowing enough space between letters. It's a fine line though because if you allow too much space WK2 will interpret that as a word space. Temporarily lowering the command speed (see command **C**) can help while you learn how the process works.

To play a message back, simply press the desired message button and the message will be sent. If you want to review the message without keying the transmitter, use the **R** (review) command. Note that review will play a pad character as di-di-dah-dah-dit and expand commands. To abort a message, press both command and PB4 pushbuttons or press and hold one of the paddles and WK2 will stop transmission immediately.

Standalone 'Two Press' Message Button Functionality

As previously mentioned above, you can trigger message 5 and 6 directly with the following sequence:
1) Press and hold PB2 for message 5 or PB3 for message 6
2) Press the command PB (you now have two PBs pressed)
3) Now release both pushbuttons and selected message will play.

Standalone Quick Tune Command

If you use the sequence outlined above but start with PB4 instead, this is way to select tune mode. This keys your transmitter until you press either paddle or both the command PB and PB4 at the same time.

Standalone Quick Serial Number Decrement

Sometimes during contest operation, a serial number has to be reissued. Since the serial number is automatically incremented when it's played, we need a way to decrement the serial number. This can be done one of two ways, using the **D** paddle command or by pressing and holding the CMD PB and then quickly pressing either paddle. In either case, after the serial number is decremented, WK2 echoes a single dit.

There is a third way to decrement a serial number, that is with a /**D** command embedded in a message. A message can be built that predecrements the serial number before sending it. For example the following two messages can be created:

```
Message 1: 5NN /N QSL ?
Message 2: /D/N QSL ?
```

Message 1 is played as the initial exchange and if the serial number needs to be resent, message 2 would be used. The second message pre-decrements the serial number before sending it. Both of these messages leave the serial number incremented after it is sent.

Standalone Embedded Message Commands

Commands can be embedded in a message. The format is the *slash* character ***DN*** (D and N sent together as one letter) followed by the desired command letter. If you want to insert the DN prosign into a message but don't want it to be interpreted as a command simply enter DN twice. Example: K1EL/1 would be entered as K1EL//1

Standalone Embedded Message Command Table

/Bnn	Set a beacon cycle time of nn seconds (nn=00 to 99). Put this at the beginning of a message to set the beacon period.
/Cn	Call message and then return
/D	Decrement serial number. (Or can use CMD PB + either paddle)
/Hn	Set HSCW speed. See table below for determining n.
/Inn	Increase Letterspace (nn=00 to 15)
/Knn	Key transmitter for nn seconds (nn=00 to 99)
/N	Play Serial Number with auto increment.
/O	Port Toggle command. This will change port 1 to port 2 or port 2 to port 1.
/P	Pause and wait for paddle entry then continue after one word space time. The pause is ended three ways 1) Paddle entry 2) Press a msg PB1-PB4, dual press to get 5 or 6 3) Dual-press CMD+PB4 to cancel.
/Qn	Set QRSS speed. See table below for determining **n**
/Snn	Set a temporary WPM (5 to 59) that is active for the duration of the message
/Un	PTT Off/On command (n=0 or 1)
/Wnn	Wait for nn seconds (nn=00 to 99)
/X	Cancel /S, /H, or /Q command
/Yn	Relative WPM Change Up (n=0 to 9) current WPM changed to WPM+**n**
/Zn	Relative WPM Change Down (n=0 to 9) current WPM changed to WPM-**n**

/1	Jump to message 1	**/2**	Jump to message 2
/3	Jump to message 3	**/4**	Jump to message 4
/5	Jump to message 5	**/6**	Jump to message 6

Rate Table for /Hn or /Qn commands

N	HSCW Rate	QRSS Rate
0	1000 lpm (200 wpm)	3 sec dit
1	1500 lpm (300 wpm)	6 sec dit
2	2000 lpm (400 wpm)	10 sec dit
3	3000 lpm (600 wpm)	12 sec dit
4	4000 lpm (800 wpm)	30 sec dit
5	6000 lpm (1200 wpm)	60 sec dit

Standalone Embedded Message Command Examples

/B60BCON DE K1EL BEDFORD NH/1 will send this beacon message in slot 1 every 60 seconds
CQ CQ DE K1EL K /W60/4 will send this message in slot 4, wait 10 secs, and then repeat
UR RST IS /P QSL will pause to allow the user to enter the RST then resume automatically. Note: Wait a full wordspace before entering after the pause or you will accidentally abort the remainder of the message.
/K05 K1EL BCON/W10/1 will key down for 5 secs, send the message, wait 10 seconds and then repeat
CQ CQ CQ DE /1 Load callsign in slot 1, this message will send CQ CQ CQ DE then the callsign.
/Y9CQ TEST CQ TEST/Z9 K1EL K1EL will play the CQ TEST portion of the message 9 WPM faster.
CQ CQ DE WB9/L12HEE/L00 K Add extra letterspace for a portion of the callsign
/H2CQ CQ DE K1EL K1EL K1EL/S15DE K1EL will send 1st part at 2000 lpm and the 2nd at 15 WPM
CQ CQ CQ DE K1I<pad>S<pad>I will send ISI with ½ space padding **(<pad> = di-di-dah-dah-dit)**
/Q1EL/1 will continuously send EL at QRS3 speed. Try to avoid inserting a space between the QRSS command and the start of text since this will cause a long delay before anything is sent. Likewise a space at the end will result in a long delay between message repeats which may or may not be desired.

Standalone QRSS/HSCW Operation

The /H and /Q buffered commands allow HSCW or QRSS strings to be sent in standalone mode. (Note that QRSS speeds are not supported in Host Mode) HSCW and QRSS strings can be aborted with either a paddle press or a Command+PB4 pushbutton press. Upon abort normal keying speed is resumed.

Standalone Preserve Settings

Once you have set up WK2 the way you like it, save the settings permanently in memory. This is done using the **P** command. Simply press the command PB, wait for the **R** and then enter ***P***. All settings will be saved. It is particularly important to do this before you change batteries since settings not stored permanently will be lost.

WK2 Standalone Tutorial

On 1st time power up, or by the restore factory defaults command, WK2 will be reset to these settings:

Operating WPM:15	Command WPM:15	Sidetone:2KHz	Weight:50%
KeyComp:0	Letterspace Adjust:0	SampleAdjust:50%	KeyMode:Iambic B
Sidetone:On	Autospace:Off	Key/PTT: Port 1	Serial Number:0001

Once WK2 has been powered up, pressing the paddle keys will generate dits and dahs both in sidetone and keyed output. Let's enter a simple command to swap the paddles. Press and hold the command pushbutton (CMD PB) until WK2 answers with an ***R***. Then, without hesitation, enter an **X** on the paddles. WK2 will answer with an **R** letting you know the command succeeded. If it did not understand the command or you are late entering the command, WK2 will respond with a question mark. After a successful command entry you'll find that the dit and dah paddles have been reversed. To swap the paddles back, enter the **X** command again.

Let's learn about the speed pot. If you turn the speed pot while sending you will notice that the sending speed will be adjusted up and down. The default range for the speed pot is 5 to 40 WPM. Let's change the range using the speed pot sweep command **S** (see page 21). Let's set a 10 WPM sweep. Press the CMD PB, wait for the **R**, and then enter **S**. WK2 will respond with an **E** telling you it's waiting for you to enter something. Enter a 1 followed by a 0. If you did it right WKUSB will answer with an **R** and now the speed range will be 5 WPM to 15 WPM. Repeat the command and use the **T** shortcut for the zero, in other words enter a 1 followed by a T. To move the entire speed pot range, use the **B** command, this sets the bottom of the sweep. If you set the bottom to 10 the range will now move from 10 to 20 WPM.

The speed used while entering commands is set directly using the **C** command. If you accidentally set the command mode to something you can use, use the Reset WK2 command on page 25 to get it back to 15 WPM.

Now try changing the keying mode. Enter the **K** command followed by a letter signifying a keying mode (see page 20). **K B** will set Iambic mode B, **K A** will set Iambic mode A **,** **K U** sets Ultimatic. Try each mode to get familiar with them. The sample adjust command allows you to tweak the paddle sensitivity to emulate your favorite keyer. Setting sample adjust to zero disables both dit and dah paddle memories.

The Weight, Keying Compensation, Letterspace, and Dit/Dah Ratio commands adjust the way Morse is generated. Read each command descriptions carefully to understand how they work.

The sidetone frequency is adjusted by using the **Z** command. Press the left and right paddles to adjust the sidetone frequency up and down, when you like what you hear, press the CMD PB to exit. With some keyers, such as WKUSB which uses a small speaker, you'll find that higher frequencies produce the loudest sidetone.

You can save your settings in WK2's internal EEPROM at any time by pressing the CMD PB until WK2 responds with an **R** and then entering a **P**.

You can toggle the keyer's output port with the **O** command, this allows you to key one of two radios from the same keyer. This saves swapping cables around when you want to move from one radio to another.

Now try entering some messages. Review the procedure for message loading on page 22. WK2 has two great features associated with messages. The first is backspace, if you make a mistake while entering a message just backspace to fix the error. The second is that the size of the message slots is not fixed, if you only use two bytes in slot one, only two bytes of message memory are used up, not an entire slot. Once you have mastered message loading you can tackle some embedded commands. An easy command to start with is the speed change command. In slot one enter: **/S10SLOW /S25FAST**. Note that the DN prosign (/) is used as the command identifier. This message will play at two different speeds. Note that after playing this message the operating speed will be returned to the original speed, in other words the speed change is not permanent.

Another set of speed related commands are **/Y** and **/Z**. These invoke a relative increase or decrease in sending rate. They are different than the **/S** command since they add or subtract from the current speed. That means you can move you speed pot around to different speeds but the relationship between the current speed and the **/Y** and **/Z** commands remains constant. For example, look at the message:
/Y5CQ TEST CQ TEST DE /Z5 WB6JJB.
The CQ TEST CQ TEST portion will be sent 5 WPM faster than the current WPM rate and then slowed back down for the callsign portion. This is helpful for difficult letter combinations. If the current WPM rate was 10 WPM the accelerated rate would be 10+5=15 WPM. If the current WPM rate was 23 WPM, the accelerated rate would

be 28 WPM. The message would have the same desired effect in both cases, to send the CQ portion of the message 5 WPM faster that the callsign portion.

Now let's compose a beacon message. In message slot 2 enter**: /B60/K05 BCON DE K1EL NH/2** When this message is played, the keyer will key down for 5 seconds and then send BCON DE K1EL NH. The **/B60** command tells the keyer to repeat the message every 60 seconds no matter how long the message itself is. To cancel a beacon, press the CMD PB or the paddle, WK2 will stop the loop and respond with an **I** to let you know something was cancelled, this is a useful signal when you cancel a beacon during dead message time.

Serial numbering is easy to use. First enter a starting serial number with the **N** command. You need to enter all four digits including leading zeroes. Next select the way you want WK2 to send 0s and 9s in a serial number. Use the **G** command for this. To play a serial number simply insert a /N command into a message. It's tricky though since the serial number is automatically incremented everytime it is played. If you want to send a serial number twice in the same message you have to decrement it after you send it like this: **UR NR /N/D /N QSL ?**

Here's a complicated set of two messages that will illustrate one way the **/P** pause command can be used.

In slot 2 enter: **CQ TEST DE K1EL K /P UR NR/N QSL?/P**
In slot 3 enter: **UR NR /D/N QSL ?**

When you press PB 2, the message in slot 2 will send CQ and then pause to let you listen for a reply. If there is no reply, hit message PB 2 again to repeat the CQ. If there is a reply, enter the station's callsign and WK2 will automatically continue, send the serial number, and then pause again. If the station needs a repeat of the serial number press PB 3 to play the message in slot 3. Since the serial number is incremented after the **/N** command in message 2, you need to pre-decrement it in order to send the correct number. Message 3 can be repeated until the station acknowledges the exchange. At that time press PB2 to start the whole process again.

The **/P** command is a three way branch. When it is encountered, WK2 will wait for one of three things to occur:

1st branch: User paddles something and the message will continue
2nd branch: User presses a message button to play a message
3rd branch: User dual-presses CMD+PB4 to cancel the message entirely.

WK2 supports two alternate sending formats. They are selected by embedded commands in a message. QRSS is extremely slow CW for VLF work, and HSCW is extremely fast CW typically used for meteor scatter QSOs.

Here are examples of each:
QRSS: **/K10/ Q2EL/2** Key down for 10 seconds followed by EL at QRSS6 rate then repeat.
HSCW: **/H3K1EL K1EL K1EL K1EL K1EL K1EL K1EL /1** Callsign in slot 1 is repeated at 3000 lpm

Document Change History:

```
01.16.2006   ste   Initial version
03.31.2006   ste   Minor Updates
12.18.2006   ste   Version 21 changes
04.14.2008   ste   Version 22 changes
10.04.2010   ste   Version 23 changes
```

Description of Version 21 changes

1) The PTT_ENABLE bit in the PINCFG register was ignored in version 20, PTT always cycled with keying. In version 21 the bug was corrected and PTT will assert automatically with keying only when PTT_ENABLE is asserted. If PTT_ENABLE is cleared the buffered PTT command can be used to cycle the PTT outputs.
2) The Buffered PTT serial command (0x18) has been re-instated. When PTT_ENABLE is clear, this command is used to control the two PTT outputs. Either PTT output can be controlled, the selection is based on which KEY bit in the PINCFG register is set. Here are some examples:
   To set PTT output 0, first set the KEYPORT0 bit in PINCFG. Then issue a Buffered PTT enable command with a parameter of zero: <0x18><0x01>
   To clear PTT output, issue : <0x18><0x00>.
   To set and clear PTT output 1, repeat the above sequence with the KEYPORT1 bit set.
   Both examples assume that PTT_ENABLE is cleared.
3) A buffered port control command has been added. The command is overloaded on the HSCW speed command (0x1D). To select Key port 0, issue the serial command: <0x1D><0x00> or to select port 1 issue <0x1D><0x01>. By combining the buffered PTT and PORT commands the PTT lines can be synchronously controlled in the midst of keying.
4) Standalone versions of the above two serial commands have been added:
   The port command is an **R** followed by either a 0 or 1 to select the desired port
   The PTT command is a **U** followed by either a 0 or 1 to either turn PTT off or on.
   Message example: **/R0TEST1 /R1TEST2 /U1TEST3 /R0/U1TEST4 /R1/U0 TEST5**

   Here is a breakdown of what the above message does:
   Select key port 0 send TEST1
   Select key port 1 and send TEST2
   Turn on PTT1 and send TEST3
   Select key port 0, turn on PTT0 and send TEST4
   Select key port 1, turn off PTT1 and send TEST5

5) A problem with saving parameters in standalone using the **P** command is fixed.

Description of Version 22 changes

1) In pre-V22 the POT RANGE parameter was not validated, if an illegal value was entered erratic behavior could result. In V22 an illegal value is ignored
2) The Speed Pot value is now averaged to reduce random speed pot changes due to noise.
3) Changes in Speed Pot are ignored during transmit. Again this reduces random speed pot changes due to induced RF. This was never proven to be a problem in V21 but it was easy to implement. The speed pot will be monitored in between active dits and dahs so that a user can still change speed while transmitting.
4) PTT release is now measured one dit time after the last element sent. In addition a tail setting of 1 means no additional delay time is inserted beyond the dit time. In other words delay equals setting minus 1. This greatly improves the QSK performance. Hang time operation has not been changed.
5) Direct (software) paddle input now operates for all modes. In v21 and earlier Iambic A and B were the only modes supported for direct paddle input.
6) V22 now operates properly when it is connected to a host application and the host PC goes into standby. This places all USB ports into a standby sleep mode. This would cause V21 to disconnect from the host and go into standalone operation. V22 will go into sleep mode in sync with the host and will wake up when the PC does. It will remain connected so that the user can continue to use WKUSB from the host application as if nothing happened.
7) Two-press pushbutton sequences are supported for standalone operation. This allows messages 5 and 6 to be accessed directly and also maps the serial number decrement command to a push button press sequence. This is described in detail on page 20 of this manual.

8) A minor bug in autospace has been fixed. In some cases an inserted autospace was a wordspace instead of a letterspace.

Description of Version 23 changes

1) A message will be stopped immediately when either paddle is pressed.
2) Message stacking: A second message can now be queued while a message is being sent.
3) Message is aborted by pressing either the left or right paddle or by pressing command and PB4 in dual-press fashion.
4) Dual-press functionality refined: To play or load message 5 or 6, first press and hold either pushbutton 2 or 3 and then momentarily press the command pushbutton.
5) Quick tune: Dual press PB4 and command pushbutton.
6) Tune is ended by paddle press or when PB4 and CMD are dual-pressed.
7) PTT is now held during space and pad characters.
8) Hang delays have been changed:
   Hang = 0: PTT is held for 1 wordspace plus 1 dit time after letter is finished.
   Hang = 1: PTT is held for 1 wordspace plus 2 dit times after letter is finished
   Hang = 2: PTT is held for 1 wordspace plus 4 dit times after letter is finished
   Hang = 3: PTT is held for 1 wordspace plus 8 dit times after letter is finished
9) Tune and Load message standalone commands have been replaced with Load Lead In time (L) and Load Tail time (T) commands. .
10) Lead in and Farnsworth are now disabled during command mode.
11) Standalone embedded port change command **/X** is now a toggle from 0 > 1 or 1 > 0.
12) QRSS and HSCW embedded command lockup on zero parameter is now fixed.
13) Analog standalone embedded commands removed.
14) Command letters Y and Z are reassigned to relative WPM change commands.
15) Greatly Improved battery life.
16) Letterspace adjustment command added for host and standalone command.
17) Relative WPM change commands added. Y is speed up and Z is speed down..
18) Speed control response improved, now updates in the midst of sending a letter
19) Changed **Q** command value ID letters to match command letters.
20) Changed operation of **B** and **S** standalone commands to adjust speed control window.
21) Set Hang command now accepts a value directly instead of cycling through settings.
22) Removed message break setting, no longer needed.
23) Dual-press CMD+PB4 needed to escape from **/P** embedded command.
24) Quick serial number decrement: Hold command pushbutton and press either paddle.

WK2 Standalone Commands – Reference Card

A	**Select sidetone on or off**	**O**	**Select output key port**
B	**Set speed control bottom WPM**	**P**	**Preserve Settings**
C	**Set command speed in WPM**	**Q**	**Query current settings**
D	**Decrement serial number**	**R**	**Review message without transmitting**
F	**Set Farnsworth Speed**	**S**	**Set speed control range in WPM**
G	**Select serial number 0/9 format**	**T**	**Load PTT Tail Time**
H	**Set Paddle Hang Time**	**U**	**Select Autospacing on/off**
I	**Set Letterspace Adjustment**	**V**	**Set Keying compensation in mSec**
J	**Set Paddle sensitivity**	**W**	**Set Key Weight**
K	**Select keyer mode**	**X**	**Exchange Paddles**
L	**Set PTT Lead In Time**	**Y**	**Set Dit/Dah Ratio**
M	**Mute Transmit, Tx sidetone only**	**Z**	**Select sidetone frequency**
N	**Load 4 digit serial number**		

Standalone Embedded Command Table

/Bnn	Set a beacon cycle time of nn seconds (nn=00 to 99). Put this at the beginning of a message to set the beacon period.
/Cn	Call message and then return
/D	Decrement serial number.
/Hn	Set HSCW speed. See table below for determining n.
/Inn	Set Letterspace increase (nn=00 to 15)
/Knn	Key transmitter for nn seconds (nn=00 to 99)
/N	Play Serial Number with auto increment.
/O	Keying Port Toggle command
/P	Pause and wait for paddle entry and then continue after one word space time. The pause is ended three ways 1) Paddle entry 2) Press a msg PB1-PB4, dual press to get 5 or 6 3) Dual-press CMD+PB4 to cancel.
/Qn	Set QRSS speed. See table below for determining n
/Snn	Set a new sending speed (nn=WPM, 5 to 59)
/Un	PTT Off/On command (n=0 or 1)
/Wnn	Wait for nn seconds (nn=00 to 99)
/X	Cancel /S, /H, or /Q command
/Yn	Relative WPM Change Up (n=0 to 9) current WPM changed to WPM+n
/Zn	Relative WPM Change Down (n=0 to 9) current WPM changed to WPM-n
/1	Jump to message 1
/2	Jump to message 2
/3	Jump to message 3
/4	Jump to message 4
/5	Jump to message 5
/6	Jump to message 6

n	HSCW Rate	QRSS Rate
0	1000 lpm (200 wpm)	3 sec dit
1	1500 lpm (300 wpm)	6 sec dit
2	2000 lpm (400 wpm)	10 sec dit
3	3000 lpm (600 wpm)	12 sec dit
4	4000 lpm (800 wpm)	30 sec dit
5	6000 lpm (1200 wpm)	60 sec dit

Rate Table for /Hn or /Qn commands

WK2 Version 23 Software and User Interface Considerations

Winkeyer Release version 23 (2.3) has a considerable number of changes and improvements which are covered in this section. There are two sections; the first will cover functional changes to standalone keyer mode. The second section will cover changes that affect host operation.

Section One: Standalone Mode Changes (No host mode implications)

1) **Message stacking**: A second message can now be queued while a message is being sent. As soon as the current message is completed the queued message will be sent.

2) **Dual-press functionality refined**: To play or load message 5 or 6, first press and hold either pushbutton 2 or 3 and then momentarily press the command pushbutton (red). When you release both pushbuttons, the desired message will be played.

3) **Dual Press Message Buttons**: Since message stacking is now supported, it is no longer possible to stop a message by pressing the command pushbutton by itself. A message is aborted by either pressing one of the paddles or by pressing command and PB4 in dual-press fashion.

4) **Quick tune command with dual-press**: First press and hold PB4, then momentarily press the command pushbutton. When you release PB4, tune mode will be enabled until either paddle is pressed or CMD and PB4 are dual-pressed.

5) **T and L Standalone Command Remapping:** 'Tune' and 'Load Message' commands have been replaced with Load Tail time **T** and Load Lead In time **L** commands. Since messages can be loaded directly and tune is set by dual-press, these commands were redundant and confusing. The new mappings provide access to settings that were not previously allowed in standalone mode.

6) **B and S Standalone Command Redefinition**: The **B** command was previously unassigned, it now is used to set the bottom of the speed pot window. The **S** command previously set the keyer's WPM rate, this was problematic since this would conflict with the speed pot setting. The **S** command is now used to set the speed pot window range.

7) **Lead in and Farnsworth are now disabled during command mode**.

8) **Port change command `/X`**: Standalone embedded port change command `/X` is now a toggle from 0 > 1 or 1 > 0. A parameter specifying the port is no longer needed.

9) **Zero QRSS/HSCW Parameter Bug Fixed:** WKUSB would lock up if a zero was given as the parameter for either the QRSS or HSCW embedded standalone commands. This is now fixed.

10) **Analog standalone embedded commands no longer supported**: These commands are not possible with the new message stacking scheme. Letters **Y** and **Z** have been reassigned as Relative WPM Change commands.

11) **New relative WPM adjustment commands for standalone messages: Y** is speed up and **Z** is speed down. Both have a single parameter value of 0 to 9 which represents the speed increase or decrease in words per minute. For example:
    `CQ TEST CQTEST /Z5K1EL K1EL/Y5 K` sends **`K1EL`** 5 WPM slower.

12) **Sleep current is greatly reduced**: When not in use, Winkeyer will go into low power standby mode (sleep). Changes in the controller's configuration at least double battery life compared to previous Winkeyer versions.

13) **Q command Changes**: Value ID letters and order now match command letters.

14) **Set Hang command**: Now accepts a value directly instead of cycling through settings.

15) **Removed message break setting**: Now that the analog readout commands have been removed, this function is no longer needed.

16) **Escape from /P command: Dual-press CMD+PB4**: Previous versions used the command button to escape from the Pause command. This is not possible with the new message stacking scheme. Instead a dual press of the command pushbutton and pushbutton 4 is now required.

17) Quick serial number decrement: Hold command pushbutton and press either paddle

Section Two: Changes that affect Host Mode

This section describes changes that have an effect on host mode operation. All of these commands also apply to standalone mode.

1) **Message break behavior change**: A message will be stopped immediately when either paddle is pressed. In previous versions, the message was not stopped until the current letter completed.

2) **Speed control response improved:** Now WPM updates are made in the midst of sending a letter.

3) **PTT functionality change**: When playing a message, or for letters sent from the host, PTT will be held during space and pad characters.

4) **Hang delays have been changed to**:
Hang = 0: PTT is held for 1 wordspace plus 1 dit time after letter is finished.
Hang = 1: PTT is held for 1 wordspace plus 2 dit times after letter is finished
Hang = 2: PTT is held for 1 wordspace plus 4 dit times after letter is finished
Hang = 3: PTT is held for 1 wordspace plus 8 dit times after letter is finished

5) **Hang mode is now independent of tail delay**: Hang Time applies only to CW sent by paddle. Tail delay applies only to CW sent by the host or by standalone message. It is no longer necessary to disable Hang Time setting controls in the GUI when Tail is set to a non-zero value. This is because the settings are independent now. (Side note: A long Tail time can be truncated early by appending a 'PTT off' buffered command **`/U0`** to the message stream)

6) **Letterspace adjustment:** New capability added for both host and standalone modes.

In **Host Mode** it is implemented as an ADMIN command, 0x15, with a single parameter calling out a letterspace scaling factor between 0 and 15. This adjustment specifies additional letterspace added in 2% increments from 0 through 30%.

In **Standalone Mode**, letterspace is adjusted using the **I** command followed by a value between 0 and 15 which represents an increase in letterspace of <**`value`**> * 2 percent. For example a value of 4 will increase letter space by 4x2 = 8%. In addition there is an embedded command, also **I**, which allows letterspace adjustment to be performed on a range of letters in a message. For example this message:

`CQ CQ de /I06K1EL/I00 K` sends **`K1EL`** with 12% added letterspace to improve readability.

Standalone letterspace setting is stored in byte 0xF of the EEPROM image. The mapping of this register is as shown.

__Bit7–Bit4	Bit3	Bit2	Bit1	Bit0
Letter space	Enable 0/9 Cut	Spare	Enable Paddle Status	Spare

7) **New ADMIN commands summary**:

ADMIN CMD	Function	Parameter
0x15	Set Letterspace	Value 0-15
0x16	Reserved	
0x17	Set 1200 Baud	none
0x18	Set 9600 Baud	none
0x19	Reserved	

Contact Information

Winkeyer2 is fully guaranteed and if you are not satisfied please return the chip for a full refund.

Please post questions on the K1EL Message Board:

http://groups.yahoo.com/group/k1el_keyers/

You can contact K1EL directly at:

Steven T. Elliott K1EL
43 Meadowcrest Drive
Bedford, NH 03110 USA

e-mail: K1EL@k1el.com

website: www.winkeyer.com