Rover Software for the PALM Organizer

Put your PDA to work figuring bearings, distances and logging contacts.

By Paul Wade, W1GHZ

A number of small handheld computer-like devices have appeared on the market. Surely there must be ham applications for them.

One of the most popular is a PDA (personal digital assistant) line from PALM Computing. It is not only useful for daily life, but also easy to program for new applications. Rather than just describe the ham programs I have developed, I will also discuss the evolution of the programs to best utilize the capabilities of the Palm platform.

Beginnings

In the summer of 1999, I acquired a Palm V PDA. The original version of the Palm was the Palm Pilot, a name that is often used for all versions. For those who haven't seen one, think of a handheld computer about the size of a chocolate bar. Some useful software is included: an address book, calendar, memo pad, "to do" list and calculator. A wide range of shareware is also available; for instance, I quickly replaced the original calculator with a scientific calculator using reverse-polish notation. Data entry is done with a stylus, by either using a stylized handwriting or tapping on a tiny keyboard with the stylus, as shown in the screen image in Fig 1.

After growing accustomed to the Palm V, I could see that this handheld device is an ideal tool for rover operation—but what about ham software? On the Internet, I didn't find anything specific, but did find a shareware *C* compiler, *PocketC*,[1] so I could develop my own. For starters, the *BD* (Bearing-Distance) program has always been essential for grid-square calculation used in VHF+ operation, and Matt, KB1VC, had already translated it into *C* code. I added some I/O (input-output) code for the Palm, and stripped the rest down to the essentials: enter two 6-digit Maidenhead locators, get back bearing and distance. The whole program requires only 4 kB of storage (total Palm V storage is 2 MB—newer models have more).

The other essential for a rover is logging. With the ARRL 10-GHz & Up Cumulative Contest approaching, I wrote a simple logging program. Again, the approach is minimalist: Record the essential information as simply as possible. Each time the program is started, you are prompted for current location, then for a call and grid for each contact. Each QSO is time-stamped and recorded, including distance in kilometers. Powering down the Palm computer does not terminate the program—it picks up instantly where you left off—so logging is quick and easy, and battery life is at least a week.

However, after the first weekend of contest use, we realized that switch-

161 Center Rd
Shirley, MA 01464
w1ghz@arrl.net

[1]Notes appear on page 51.

ing back and forth between *BD* and the logger was inconvenient. Matt suggested that a better approach would be to incorporate *BD* into the logger: Enter the grid, get the heading then enter the call upon successful completion of the contact. The next versions operated in this manner. All this sophistication bloated the program up to 7 kB of storage.

I originally made two versions of the logging program: *10G_LOG* for 10-GHz-only operation, and *LOG_CUM* for multiband operation. The logged fields are band, date and time, calls, grids and distance, as needed for the ARRL contest. Rather than invent a fancy database, the log is kept as a memo in the Palm built-in *MemoPad* application, writing one line per QSO. Operations like duping, formatting, QSO points and totals are best done after the contest on a desktop PC system. The Palm has a standard feature that transfers information to the desktop system via a serial or USB port and keeps both systems current with a "Hot-Sync" button.

Evolution

The simple logging program required text entry of all data, just as if the user had a keyboard. A few users helped by testing several versions through bug fixes and improvements. Finally, Simon, GM4PLM, suggested that it would be easier to select the band from a menu rather than enter it for each QSO.

Simon's suggestion sounded reasonable, so I looked for ways to implement it. Accompanying the *PocketC* compiler are some sample programs, including a tic-tac-toe game, played by tapping the desired box with a stylus. I modified the code to display a band in each box and detect which band was tapped, then included this function in the logging program. I'm not a professional programmer, but I do know that modifying working code for new uses improves programming productivity.

A short digression about *C* programming might be in order. *C* is a compiled language, which must be translated (compiled) into machine language before it can be executed, or run on the computer. *BASIC*, on the other hand, is often an interpreted language—the computer uses a *BASIC* interpreter, translating each time it runs a program. The compiled code typically runs much faster, which may not be noticeable on a fast Pentium PC, but the processor in a handheld device is much slower for reduced power consumption.

Good programming in any language uses a modular approach, using functions (in *C*) or subroutines (some in other languages) for specific operations. I wrote the logging program as a set of modular functions, so that operations might be enhanced by replacing the appropriate function. For instance, incorporating *BD* into the logger was a simple matter of adding a call to the `bd()` function. The compiler does the rest! Later, we will see that making several different versions of the logging program for different contests is done by creating different `main()` functions that call the appropriate functions for that version—the compiler uses those functions and ignores the rest.

Table 1 shows the *C* code for the `main()` function of my general-purpose log. All the text on a line after "//" are comments, always a good programming practice to explain what is happening; even the original programmer forgets eventually. As you can probably see, this routine doesn't do much, just starts the program, finds the desired log in the memo pad, and then calls another function, `AddVHFUpLog_BD(mycall)` to do the actual logging.

Another useful programming practice is to generalize functions by using parameters. In the case of the tic-tac-toe board, we can generalize it from 3×3 boxes to *X*-by-*Y* boxes; *X* and *Y* are parameters. I included additional parameters for the size of the boxes and position on the screen, so that I could use one function to create any number of menus. Table 2 shows the function that draws the lines on the screen to create the boxes.

Finally, when I select a grid by tapping it with the stylus, a few Palm-specific instructions find the coordinates of the tap on the screen. Then the coordinates are converted to an offset—the number of grids in each direction away from my current grid, as shown in Table 3.

I hope these bits of code give you some of the flavor. If you like it, look at the rest of the source code in 0303LOGS.ZIP;[2] I've tried to provide enough comments to make it readable. Perhaps you will be inspired to improve on it and personalize your log.

More important than the implementation language and detail is the design philosophy. I bought and read

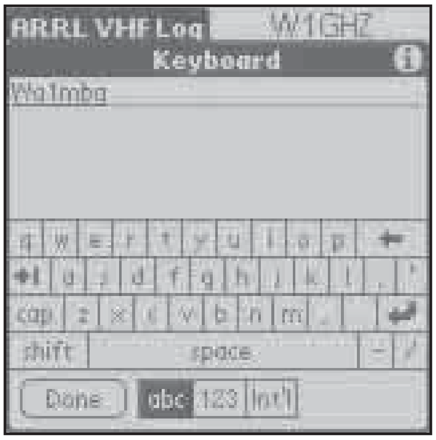


Fig 1—The keyboard pattern used by the Palm V for text input.

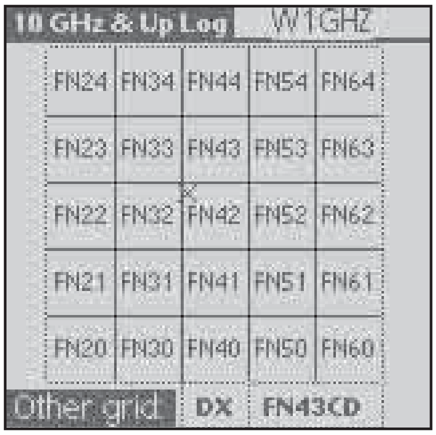
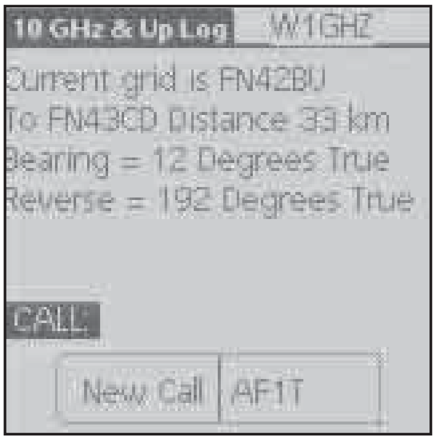


Fig 2—A screen capture of the 5×5 box matrix used for nearby Grid input.


10 GHz & Up Log W1GHZ

Current grid is FN42BU
To FN43CD Distance 33 km
Bearing = 12 Degrees True
Reverse = 192 Degrees True

CALL:
New Call AF1T

Fig 3—Results of calculations performed by *BD,* along with a prompt for a new call sign.

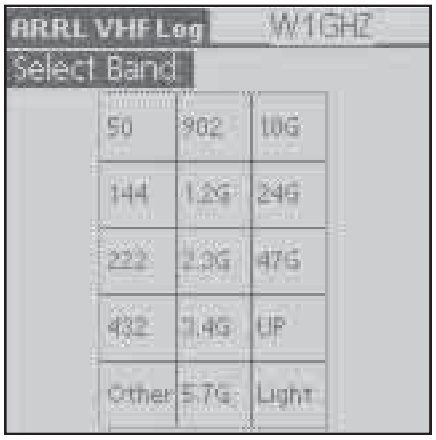


Fig 4—A single stylus tap can enter bands up through 47 GHz. Higher bands require a text entry.

Table 1—*C* code for the main()function of the General-Purpose Log

```
// VHF & Up
// VHF & Up log
// by Paul Wade W1GHZ
// version 1.0 December 1999
// precalculate grid map and store
// version 0.9 December 1999
// HISTORY:
// from
// Log Cum
// simple log for the Microwave Cumulative Contest
// by Paul Wade W1GHZ
// version 0.6 November 1999
// select grid and band by tapping on map
// version 0.5 September 1999
// don't write null calls
// version 0.4 August 1999
// ask for grid, return heading, then ask for call
// version 0.3 August 1999
// fixed GMT offset saving


// GLOBAL VARIABLES:
int Zoffset;  /* global GMT offset */
float Distkm, Baz;  /* ugly globals to avoid pointers */
string adj_grid[25];  /* precalculate adjacent grids for map */

// files with called functions
include "GridFunc.c"
include "BD_full.c"
include "Gridpeck.c"
include "BandPeck.c"
include "LogFunc.c"

// note: functions lack type in PocketC

StartVHFUpScreen()
{
// text(x,y,string) is a graphics mode function
// screen is 160x160, x,y=0,0 is upper left
   text(110,0,"v1.0");
   text(0,20,"VHF & Up Log Program");
   text(0,35," by W1GHZ 1999 ");
   text(0,50,"General VHF+ Logging");
}

main()
{
   string mycall;
   clear();
   graph_on();          // use graphics mode
   clearg();

   title("VHF & Up Log");   // at top of screen

   textattr(2,1,0);

   StartVHFUpScreen();

   GMTadjust();         // offset for GMT

   StartVHFUpScreen();

   mycall = gets("My Call:");   // gets() creates a text input box

   mycall = strupr(mycall);

// look for a MemoPad record starting with the string 'mycall'
   if (mmfind(mycall))          // found, use it
     {
     clearg();
     text(100,0,mycall);
     text(0,20,"Log file found");
     AddVHFUpLog_BD(mycall);    // add entries to existing log
     }
   else if (mmnew())            // not found, start one
     {
     NewHeader(mycall);              // create the new log memo
     clearg();
     text(100,0,mycall);
     text(0,20,"New Log file started");
     if (mmfind(mycall))   // make sure the log is there
       AddVHFUpLog_BD(mycall);      // add entries to the log
     }
   else                         // can't find or create
     {
     clearg();
     textattr(2,2,0);
     text(0,35,"Couldn't open log file ");
     textattr(2,1,0);
     beep(3);
     }

   mmclose();                   // finished: close the MemoPad
}
```

a book on Palm programming,[3] which helped me understand how to use the Palm effectively. One important concept is to consider the Palm as an extension of the desktop display, rather than as a standalone computer; by relying on the desktop to post-process the log, I was heading in the right direction. Another concept is to minimize the amount of data input required; my text-based logging had already proven slow and tedious.

I thought about VHF+ contest operating: How could I make logging faster and simpler? First, unless a station is in a remote location, many of the contacts are in nearby grid squares; a menu of adjacent grids would be much easier than entering each one. Next, we frequently "run the bands" with a station, working him or her successively on all bands we have in common; clearly, the grid and call don't change, so storing these from the previous contact is important. Since we use rather sharp beam antennas, another frequent occurrence is to work several stations in the same grid or portable location; again, storing the grid location is important. Finally, we occasionally hear or enter data wrongly initially, so easy editing of a QSO before confirming it for the log is important.

Now I had a design philosophy: Make these frequent entries easy, with a

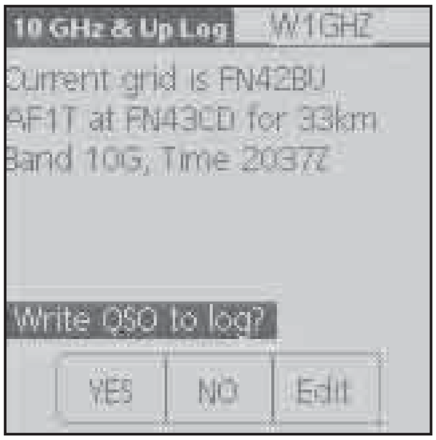

10 GHz & Up Log W1GHZ
Current grid is FN42BU
AF1T at FN43CD for 33km
Band 10G, Time 2037Z
Write QSO to log?

Fig 5—Once the band is selected, the entry information is displayed for confirmation before it is stored.

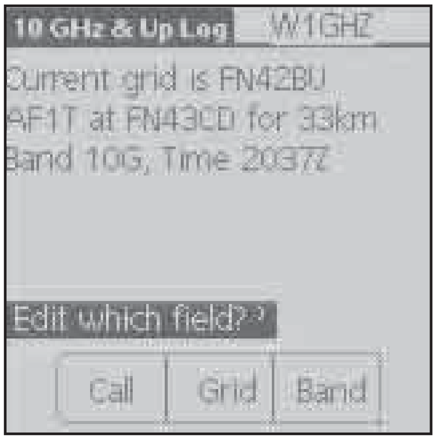

10 GHz & Up Log W1GHZ
Current grid is FN42BU
AF1T at FN43CD for 33km
Band 10G, Time 2037Z
Edit which field?

Fig 6—Tapping "Edit" in the screen of Fig 5 sends the user to this editing screen.

Table 2—A Function that Draws Lines on the Screen to Create Boxes

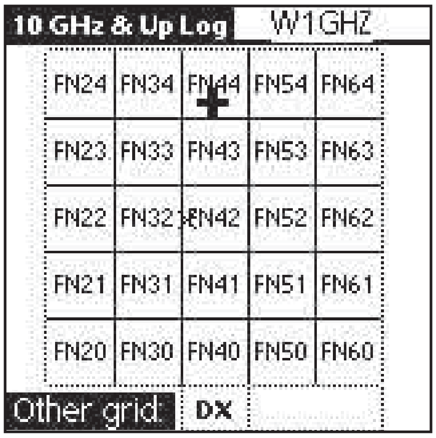
```
Outlineboxes(int xbox,int ybox,int xstart,int ystart,int xincrement,int yincrement)
// draws a set of boxes on the screen
// number of boxes is xbox by ybox
// xstart and ystart are lower left corner
// xincrement and yincrement are size of box

  {
  int m;

// vertical lines
  for (m = 1; m <= xbox; m++)
    line(1,(xstart + (m * xincrement)),ystart,
          (xstart + (m * xincrement)),(ystart + (ybox * yincrement)));
// horizontal lines
  for (m = 1; m <= ybox; m++)
    line(1,xstart,(ystart + (m * yincrement)),
         (xstart + (xbox * xincrement)),(ystart + (m * yincrement)));
// Draw the gray border
  frame(2, xstart,ystart,(xstart + (xbox * xincrement)),(ystart + (ybox * yincrement)),1);
  }
```

Table 3—Tap Coordinates are Converted to an Offset from the Current Grid

```
stylus_event_example()  // code snipet
// find the peck
// Wait for a stylus event
    {
     waitp();          // waits for pen event
     x = penx();       // returns pen x location
    y = peny();   // returns pen y location
  // which box was it in?
     x = ((x-xgstart)/xgincrement) - 2; //human readable direction
     y = ((y-ygstart)/xgincrement) - 2;
   // calculate grid using x and y
   // return grid
    }
```

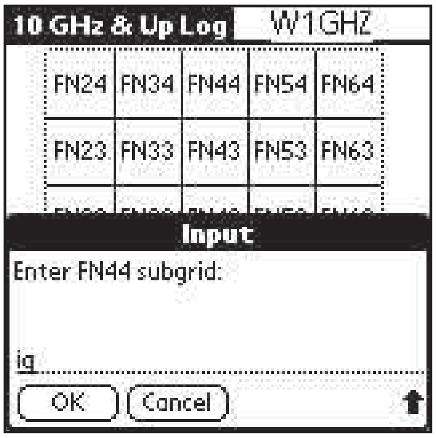


10 GHz & Up Log W1GHZ

FN24	FN34	FN44	FN54	FN64
FN23	FN33	FN43	FN53	FN63
FN22	FN32	FN42	FN52	FN62
FN21	FN31	FN41	FN51	FN61
FN20	FN30	FN40	FN50	FN60

Other grid: DX

(A)



10 GHz & Up Log W1GHZ

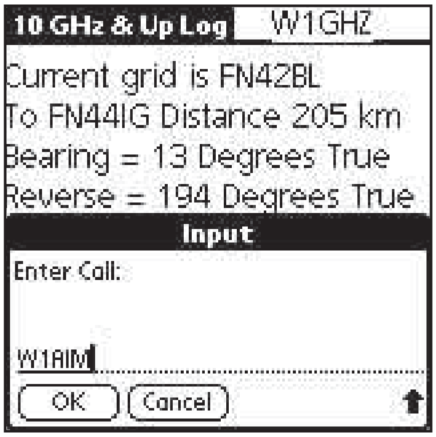
FN24	FN34	FN44	FN54	FN64
FN23	FN33	FN43	FN53	FN63

Input

Enter FN44 subgrid:

ig

(B)



10 GHz & Up Log W1GHZ

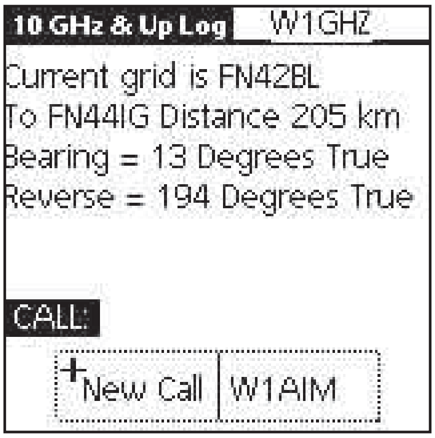
Current grid is FN42BL
To FN44IG Distance 205 km
Bearing = 13 Degrees True
Reverse = 194 Degrees True

Input

Enter Call:

W1AIM

(C)

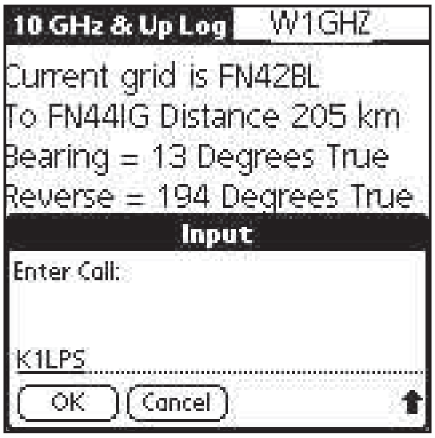


10 GHz & Up Log W1GHZ

Current grid is FN42BL
To FN44IG Distance 205 km
Bearing = 13 Degrees True
Reverse = 194 Degrees True

CALL:

(G)



10 GHz & Up Log W1GHZ

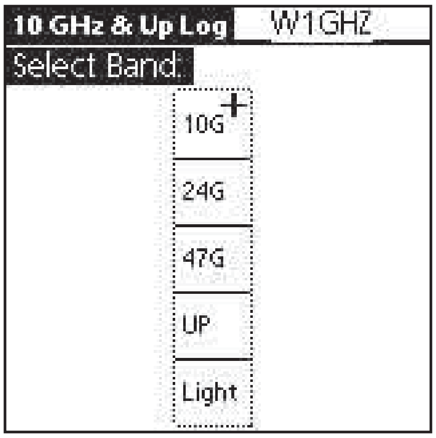
Current grid is FN42BL
To FN44IG Distance 205 km
Bearing = 13 Degrees True
Reverse = 194 Degrees True

Input

Enter Call:

K1LPS

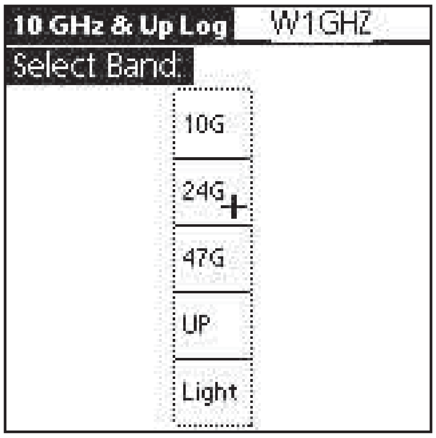
(H)



10 GHz & Up Log W1GHZ

Select Band:

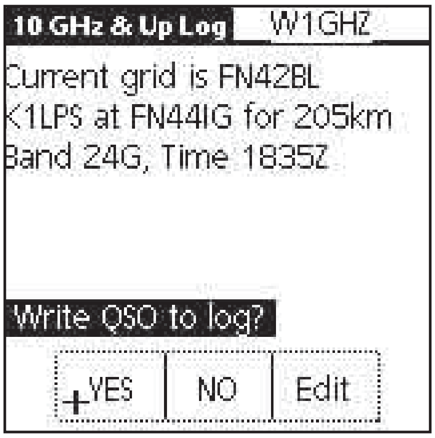
(I)



10 GHz & Up Log W1GHZ

Select Band:

(M)



10 GHz & Up Log W1GHZ

Current grid is FN42BL
K1LPS at FN44IG for 205km
Band 24G, Time 1835Z

Write QSO to log?

(N)

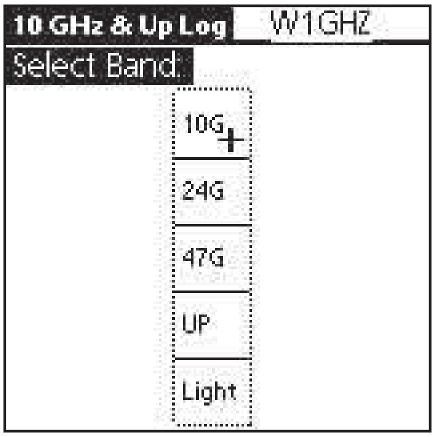
Fig 7—A typical sequence of actions to log several contacts (see text).

single tap of the stylus if possible, and use text entry only for unique data. The first entry of a call is certainly unique, but successive contacts with that station on other bands are not.

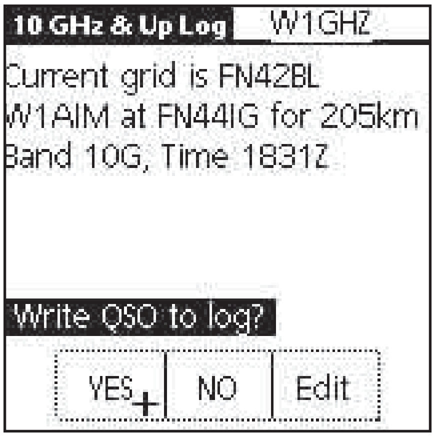
The obvious implementation for this design philosophy is the generalized tic-tac-toe menu. For grid selection, a 5×5 box menu allows selection of two concentric rings of grids in any direction, as shown in the screen capture of Fig 2. Note the small "x" at my actual 6-digit grid location in the central box, allowing quick visualization of approximate beam heading. The 4-digit grid (or the previous grid, in the lower-right box) is chosen with a stylus tap, then the final two digits are added as text. For grids outside the menu area, a "DX" choice prompts for text entry.

When the grid is complete, the bearing and distance are calculated and displayed, as in Fig 3, along with a prompt for the call. If the previous grid was chosen, a small menu gives you a choice: a new call or the previous one with a single tap.

After the call is entered, another menu, shown in Fig 4, allows choosing the band with a single tap.



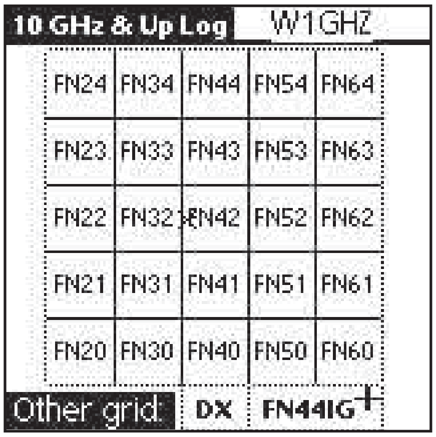
(D)


10 GHz & Up Log W1GHZ

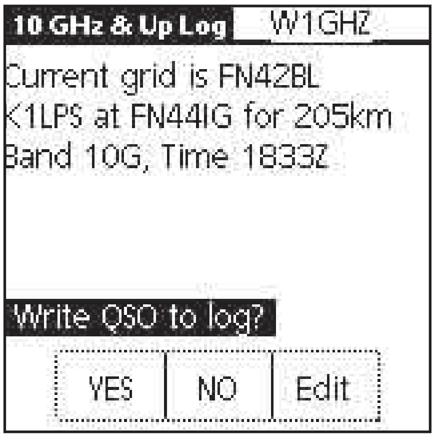
Current grid is FN42BL
W1AIM at FN44IG for 205km
Band 10G, Time 1831Z

Write QSO to log?

(E)



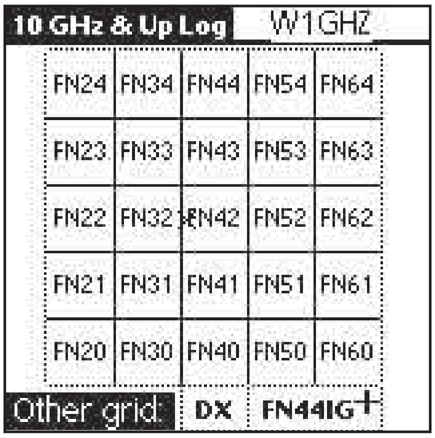
(F)


10 GHz & Up Log W1GHZ

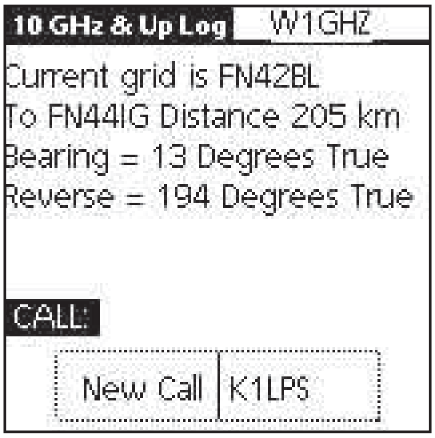
Current grid is FN42BL
K1LPS at FN44IG for 205km
Band 10G, Time 1833Z

Write QSO to log?

(J)



(K)


10 GHz & Up Log W1GHZ

Current grid is FN42BL
To FN44IG Distance 205 km
Bearing = 13 Degrees True
Reverse = 194 Degrees True

CALL:

(L)

However, for bands above 47 GHz, text entry is required—stations on those bands are probably willing to tolerate the extra effort!

Once the band has been selected, we are ready to log the contact. The data is displayed for confirmation, as shown in Fig 5. Our final menu is to write the QSO to the log: "Yes", "No" and "Edit" are the choices. The first two are obvious, while "Edit" leads to still another menu, shown in Fig 6; each choice allows re-entry of the appropriate field, followed by another chance for confirmation. When the QSO is written, the current date and UTC are automatically added.

A random contact can thus be logged with three taps of the stylus plus text entry of the call and last two digits of the grid square. Contacts based on the previous one are even easier to enter; the best case, running the bands with a station, takes four taps per QSO.

Example

To see how easy logging has become, look at Fig 7; it is a cartoon sequence of log screens for several successive (hypothetical) contacts. Microwave contacts usually begin with some liaison on a lower band, usually 2 meters. W1AIM has informed me that he is portable in grid FN44ig; this is Mt Washington, New Hampshire, so he is probably cold and wet and will appreciate my logging the contact quickly! I look at the grid map on my Palm and tap on FN44 (I've added the large cross to the picture to show the location of a tap). Up pops a text-input window, shown in the next panel

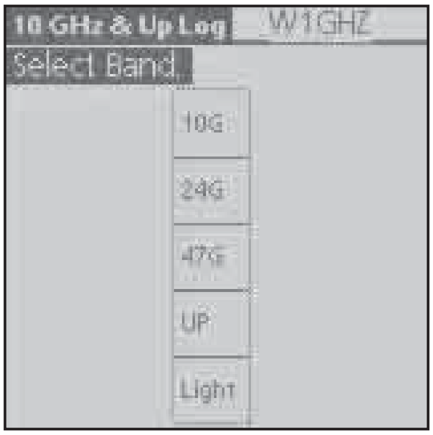


Fig 8—The Band Select screen for the ARRL 10-GHz and Up Cumulative Contest.

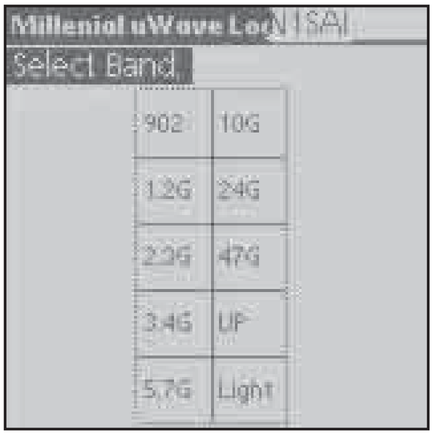


Fig 9—The Band Select screen for the NEWS Millenial Cumulative Microwave Contest.

Fig 10—The Palm-V menu screen showing all ham applications on W1GHZ's Palm V.

Fig 11—W1GHZ operating the 2001 ARRL 10 GHz & Up Cumulative Contest from the top of Mt. Wachusett, Massachusetts, and logging on his Palm V. Photo by George Caswell, W1ME

Fig 12—W1GHZ operating the 2002 ARRL 10 GHz & Up Cumulative Contest from Block Island, Massachusetts, and logging on his PALM V.

(B). I add the "ig" sub-square to the grid using a stylus to write in the text-input box at the bottom of the screen (not shown in these cartoons). Then I tap on "OK," and the Palm calculates the beam headings and distance, shown in the next panel (C) and prompts for a call.

Once we have completed a successful exchange, I enter the call in the text-input box and tap "OK." A band menu pops up; I tap "10G" and the entry is complete. The information is displayed for a final check, shown in the fifth panel (E). If I tap "Yes," then the QSO is written to the log; if I made a mistake, a quick tap on "Edit" allows on-the-spot correction.

While I am logging this contact, another station is "tail ending" on 10 GHz. Since dish antennas are very sharp, he is probably in the same grid. After we complete a successful exchange, I can log this QSO quickly: The previous grid is in the lower right corner of the grid map, shown in panel F, so I can repeat it with a single tap. The next screen (panel G) again displays the bearing and distance, and asks if this is a new call—if it were the same call on a different band, a single tap would suffice. However, this is a new call, so I tap "New Call," then enter the call on the next screen (panel H). Then a tap on the band menu (panel I) and a "Yes" for confirmation (panel J) writes this QSO to the log.

Meanwhile, Larry has suggested moving to another band, 24 GHz, while we have the antennas aligned on each other. Assuming this QSO is also successful, all the information is the same except for the band, so it can be logged with one tap on each of four successive screens, each shown in a panel.

The cartoon sequence shows how easy logging can be: three QSOs logged with 14 taps of the stylus, one per screen, and only 12 characters actually entered. With a little practice, it becomes easy enough that logging doesn't require your full attention. This is important, since there are a few other things to do simultaneously: copy the contest exchanges, listen for talk-back on the liaison radio, keep stuff from blowing away and answer questions from the tourists. The last is important—we are also public relations ambassadors for Amateur Radio!

Other Versions

The previous screen examples are for a general-purpose VHF+ log, not for any specific contest. I have made four other versions for specific contest logging:

10G & Up: for the ARRL 10-GHz-and-Up Cumulative Contest. The major difference is the limited choice of bands, as shown in Fig 8.

Log 10GHz: for 10-GHz-only operation in the ARRL 10-GHz-and-Up Cumulative Contest, since most operators today operate on a single band. The band is logged automatically.

Millenial: for the Millennial Cumulative Microwave Contest, sponsored by the N.E.W.S. Group, a yearlong microwave contest. The band menu for the microwave bands is shown in Fig 9. Contest rules may be found at **www.newsvhf.com**

ARRL VHF: for the ARRL VHF Sweepstakes and QSO Parties, as well as the ARRL UHF Contest. Since only four-digit grid squares are required and VHF antennas aren't as sharp, a slightly different paradigm is used here: The first prompt is for the call, but with an option to run *BD*. A one-tap grid selection is enough for nearby grids except when running *BD*—the additional digits are needed for accurate calculation.

The Palm menu screen for my ham applications is shown in Fig 10, with icons for all the different versions of logging, *BD_Palm*, and two shareware applications, *RiseSet* and *COMPASS*, which are described below. A tap on an icon will start the program. Figs 11 and 12 show me enjoying the fruits of my labor.

Limitations

These logging programs are aimed at roving and contest operations where small size and convenience are more valuable than the power of a laptop computer and the speed of keyboard entry. One hidden limit is the maximum size of a Palm memo-pad record—roughly equivalent to 60 QSOs. In some parts of the country, this would be perfectly adequate, but not in high-activity regions.

There are at least two solutions:

Occasionally renaming the memo pad record and starting a new one.