SEIKO
UC-2000 / UC-2200
User's Manual

This manual is not a copy of the original one. I wrote it by myself because I had none. This is the result of hours of testing using my global knowledge in programming and, most of all in Basic Programming.

Guillaume Tello, 2008.
guillaume.tello@orange.fr

Table of contents

Chapter 4: ROM Applications

Chapter 5: Calculator

Chapter 1 :
Use as a Watch

The Seiko UC-2000 is a watch driven with four buttons :

1. Split, Reset, **Set**, Scroll Up
2. **Mode**
3. **Transmit**
4. Start, Stop, **Select**, Scroll Down

For each, its usual name appears in **bold**, that's the one we'll use.

The normal display looks like this:

```
mm-dd
day name
 HH:MNse
```

MODE:

Parses the differents screens and options of the watch, if no application is loaded, you get:

Date/Time	MODE	MemoA	MODE
MemoB	MODE	Alarm setting	MODE
StopWatch	MODE	Time&date setting	
	MODE	Back to Date/Time display	

If an application is loaded, it replaces MemoA and MemoB:

Date/Time	MODE	ApplName	MODE
Alarm setting	MODE	StopWatch	MODE
Time&date setting	MODE	Back to Date/Time display	

Time and Date setting:

- With MODE go to Time & Date setting (blinking seconds)
- With SELECT move to the data you want to modify
- With SET change its value
    - particular case of the seconds
        - SET with seconds<30 rounds to the current minute
        - SET with seconds>30 rounds to the next minute
- With MODE exit to the normal display

Alarm Setting:

- With MODE go to Alarm setting (blinking ON or OFF)
- With SELECT move to the data you want to modify
- With SET change its value
- With MODE exit to the normal display

You may not hear your alarm if the bell character is not present on the normal display! To set it, push both SET and SELECT buttons on the normal display.

StopWatch:

- With MODE go to StopWatch
- If "STOPWATCH" or "STOP" is displayed then:
  - with SELECT you start the chronometer, "RUN" appears
  - with SET you reset the chronometer
- If "RUN" is displayed then:
  - with SELECT you stop the chronometer and "STOP" appears
  - with SET you copy the current time to the upper value, "SPLIT" appears and the lower chronometer goes on
- With MODE exit to the normal display

For how to use the Memo or the Applications, see next chapters.

Chapter 2:
Memo A and Memo B

Startup

- Mount the watch (UC-2000) on the Transmission Circuit
- Push the TRANSMIT button, you should see:
  - "TRANSMIT"
  - STAND-BY
- Switch the keyboard (UC-2200) ON and push the M-A button, you should see(*):

```
1. EDIT
2. COPY
3. BACK UP
4. LOAD
5. COPY/Bup
```

- Or push the Shift + M-B button, you should see(*):

```
1. EDIT
2. COPY
```

To select the line you want, use the **down and up arrows** and then **ENTER**.

(*) *If a Basic program is present in memory, you are asked wether or not you want to delete it and turn the memory into Memo.*

Preparing a Memo with the Keyboard

1) EDIT:

Allows you to edit the memo. You can use the whole keyboard, the arrows to navigate, the function keys.

When you have finished:

- if you use STOP key all modifications are cancelled
- if you use Shift+F10 (End) modifications are accepted

2) COPY:

Prints the whole Memo:
if empty, you get Memory Empty
if filled, you get:

```
** COPY (Memo A) **
     1: line1
     2: line2…
*****   END   *****
```

3) BACK UP *(not available for Memo B)*

Saves a copy of the Memo in the keyboard's memory.

4) LOAD *(not available for Memo B)*

Restores the Memo with a previous back up.

5) COPY/Bup *(not available for Memo B)*

Prints the backup memo.

Reading memos from the watch

- With MODE go to MemoA or MemoB
- With SELECT enter in
- With SET and SELECT scroll up and down the Memo
- With MODE exit to the normal display.

Chapter 3 :
Basic Programming

Startup

- Mount the watch (UC-2000) on the Transmission Circuit
- Push the TRANSMIT button, you should see:
  - "TRANSMIT"
  - STAND-BY
- Switch the keyboard (UC-2200) ON and push the BASIC button, you should see:
  - Copyright
  - (c)1983 by
  - Microsoft
  - 2922 bytes
  - free
  - Ok
- Bytes free are not displayed if there is yet a Basic program in memory.

Then you can enter a program (with line numbers) or perform immediate commands. **To exit from the BASIC use the F10 key (Shift+F5).**

Conventions :

li	line number (immediate value, not an expression)
ni	immediate integer number
n	number or numeric expression used as an integer
x	number or numeric expression
str	string or string expression
[…]	optional part

All commands and instructions can be written in upper or lower case. When listed, they all will appear in upper case.

Direct mode commands

`RUN [li]`

Runs the program [starting at line li].
F5 can be used too.

`EDIT li`

Edition of line li, F2 is a shortcut for EDIT.
Arrows can be used, F1 and F6 to insert/delete, F9 to clear the line.
Validate your modifications with RETURN.
To cancel the modifications, clear with F6.

LIST [li]

Display the listing of the program using the line number order.

With li specified, starts from this line til the end.

To stop the listing, press STOP, a “Break” will appear.

LLIST [li]

Same as LIST but printed on paper instead of displayed on screen. There, listing is faster because the printer is faster and uses 20 characters per line instead of 10.

NEW

Erases the whole program and variables from memory. You go back to the 2922 bytes free.

CLEAR

Erases all variables values and definitions (those declared with DIM). The program remains in memory.

CONT

If a program is stopped with STOP (instruction or key), you can go on with it using this command. Program should not be modified, variables can.

Variables

There are two kinds of variables: real numbers and character strings. They are distinguished with a final "$" for strings. The variable name uses two alphanumeric characters, if you use more, only the first two will be signifiant.

For example:

A, X0, KY are real number variables.

A$, X0$, KY$ are string variables.

And COUNT is the same variable as CO.

A real uses 4 bytes and can range from 9.5E-39 to 1.7E+38 with approximately 7 digits precision. A string uses a variable amount of bytes, depending of its length.

Then you can build multidimensinal arrays of reals or strings using the DIM instruction:

`DIM var(n1 [,n2,..,nmax] )`

declares var as an array, it seems that max=5.

DIM A(25): A is an array of 26 reals from A(0) to A(25).

DIM TX$(1,9): TX$ is an array of 20 strings from TX$(0,0) to TX$(1,9).

`[LET] var=expression`

computes the expression and stores it into var. LET is optional.

For example A=5*X-2

Z$(I)="-"+LEFT$(K$,3)

User Defined Functions

You can declare functions and call them by their name within a program. Only real functions can be used and they have only one real parameter. They are declared with:

`DEF FNvar(x)=..expression of x..`

Declares FNvar as a function, then to use it, as a numeric value, just write FNvar(x).

For example:

```
10 DEF FNDB(X)=X*2
20 PRINT FNDB(7)          will print 14
30 A=SQR(FNDB(8))         will store 4 in A
```

X is a local variable and doesn't affect the value of X if present in the program.

Instructions

Most instructions can also be used in direct mode.

`:`

Instruction separator if you want to put more than one on the same line.

`CLS`

Clears the screen and the cursor goes to the upper left.

BEEP

Outputs a sound.

REM text

Inserts a remark in the program. The whole remaining line is considered as part of the remark, so REM hello: PRINT A will never print A.

FOR var=x TO x' [STEP x'']
instructions
NEXT var

initialises var to x and repeats the instruction block until var>x'.

If no STEP is specified then, STEP=1 to increment var at each loop.

If x''<0, then loop until var<x'.

GOTO li

jumps to the specified line.

ON n GOTO li1 [, li2…., limax]

computes the integer value of n, and, if n=1 jumps to li1, if n=2 to li2 etc.

if n<1 or n>max then no jump is performed and go to the next instruction.

GOSUB li
RETURN

jumps to the specified line containing a subroutine that ends with RETURN.

When this last is reached, executions goes on after the GOSUB.

ON n GOSUB li1 [, li2…., limax]

Same as ON..GOTO but with GOSUB.

IF condition THEN instructions
IF condition THEN li

If a condition is true (different from zero) then the following instructions are executed, or jump to the specified line number.

If the condition is false (equal to zero) then executions goes on to the next line.

No ELSE instruction is available.

See at logical operators to know how to write a contition.

PRINT [str or x[, or ; [etc…]]]

Prints the list of numeric or strings values.

If separated by ";" they are displayed next to the previous, if separated by "," a larger zone is used between values. Each numeric value is preceeded by a space if positive or by a "-" if negative.

You can omit the ";" separator between an immediate string and a value. For example:

PRINT "X=";X;"Km" can be compacted as PRINT "X="X"Km"

If the expression list ends with a "," or ";" this wil determine the starting point for the next PRINT instruction, else a new line will be used.

A single PRINT fills a blank line.

LPRINT [str or x[, or ; [etc…]]]

The same as PRINT but for the printer!

INPUT [«message» ;]var1[,var2,…]

Display the message, if present, display "?" and ask for the values of each variable. You must separate the values with ",". For example:

INPUT A,F$

?

If you type 3.7,yeah then A=3.7 and F$="yeah"

If you give more values than expected you'll get the warning "Extra ignored".

If you give less values than expected, a new line with ?? appears asking for more.

If you give a string instead of a real, you'll get the warning "Redo from start".

DATA value_list

READ var1 [,var2,…]

RESTORE [li]

DATA inserts a list of immediate values (reals or strings, but not expressions) in the program. READ allows you to parse them (as an INPUT but without a human intervention) and RESTORE tells from wich DATA line the next READ should pick its values.

For example:

```
10 RESTORE 100
20 INPUT "Digit=",D
30 READ A,A$
40 IF A<0 OR A=D THEN 60
```

```
50 GOTO 30
60 PRINT A$:GOTO 10
100 DATA 0,ZERO,1,ONE,2,TWO,3,THREE
110 DATA 4,FOUR,5,FIVE, 6,SIX,7,SEVEN
120 DATA 8,EIGHT,9,NINE,-1,UNKNOWN
```

Note: Line 10 could be RESTORE alone as the very first DATA is in line 100.

STOP

CONT

Stops the execution, displays Break in line_number and Ok+cursor.

If you don't modify the program, you can start it where it stopped with CONT.

END

Stops the program.

Functions

FRE(0)

Returns the number of free bytes, 2922 at start. The argument is ignored, then FRE(-9.3) is the same.

RND(x)

Returns a pseudo-random number from in [0;1[.

If x>0, returns the next random number.

If x=0, returns the last random number.

If x<0, initilalizes a new serie of random numbers according to x.

For example, if you wish your program to always use the same random number serie, then add A=RND(-8) at the beginning of your program.

SQR(x)

Returns the square root of the positive value x.

INT(x)

returns the highest integer lower or equal to x.

INT(7.9) = 7

INT(10) = 10

INT(-4.1) = -5

If you prefer to get INT(-4.1)=-4, you can define the following function:

DEF FNI(X)=SGN(X)*INT(ABS(X))

ABS(x)

Returns the absolute value of x.

SGN(x)

Returns the sign of x in this way:

if x>0 then SGN(x)=1

if x<0 then SGN(x)=-1

if x=0 then SGN(x)=0

COS(x)

SIN(x)

TAN(x)

ATN(x)

Returns the cosine, sine, tangent, and reverse tangent of the x argument.

The angle is in radians.

EXP(x)

LOG(x)

Returns the exponential or logarithm of x (base is e=2.71828…).

INKEY$

Returns the current key pressed else a null string.

Example:

```
10 A$=INKEY$: IF A$="" THEN 10
```

This waits for a keypress, then A$ can be used to determine wich action to perform.

ASC(str)

CHR$(x)

ASC: Returns the ASCII code of the first character of the string.

CHR$: Returns a one character string corresponding to the x ASCII code.

Why does PRINT CHR$(165) lead to a System Error?

VAL(str)

STR$(x)

VAL: Returns a real described in the string.

STR$: returns the string descibing the real x.

LEFT$(str,n)

Returns a string with the n leftmost characters of str.

RIGHT$(str,n)

Returns a string with the n rightmost characters of str.

MID$(str,n1,n2)

Returns a string with n2 characters starting at position n1 in str.

Operations

Maths operations:

+ - * / () ^

In an expression you can use the four operations combined with parenthesis:

A=3*(15+7/(5-3)) is $A = 3 \times \left(15 + \frac{7}{5-3} \right) = 55{,}5$

Yan can use the "power" operator:

2^3 this is $2^3 = 8$,

the exponent can be real:

10^3.7 this is $10^{3{,}7} = 5011{,}87$

and even negative:

6^-3 this is $6^{-3} = 4{,}62963E-03$

Logical operations:

= > < >= <= <>

Those 6 relational operators return either -1 (True) or 0 (False)

Examples:

PRINT 3=2 returns 0

A=7: PRINT 2*A=14 returns -1

AND OR NOT

Those three operators are used to combine several conditions.

Example:

IF (A>0)AND(A<>10) THEN 20

jumps to 20 if A is positive and not equale to 10.

But they also perform a bit to bit logical operation:

6=110b and 3=11b

6AND3 = 010b = 2

6OR3 = 111b = 7

NOT6=111…11001 = -7

XOR doesn't exist, definition is

A XOR B = (NOT A AND B) OR (A AND NOT B)

But this requieres a too long expression, if A and B come from two logical comparisons (thus returning only -1 or 0) you can use A<>B example:

IF (X=7)<>(Y=2) then 20

will jump to line 20 only if exactly one contidion is TRUE.

Error Messages

Clear messages:

`Redo from start`
: Bad response to an INPUT statement

`Extra ignored`
: Warning if you enter too many values to an INPUT

Encoded messages:

Those appear in this format:
`?XY Error`
Where XY is a two character code.

`/0` Division by zero

`BS` Bad Subscript (use of A(100) with DIM A(50))

`DD` Maybe use inside a program of a command that requires direct mode.

`FC` FunCtion, domain error, example SQR(-1)

`ID` Maybe use of a function in direct mode that is reserved for programming

`OD` Out of Data, READ has reached the end of DATA statements

`OM` Out of memory, example DIM A(1000)

`OS` Out of Space? string too long ?

`OV` OVerflow, a number exceeds 1,7E+38

`SN` SyNtax error, example a bad spelling PRIN A instead of PRINT A

`TM` Type Mismatch, usr of a string where a real is expected and vice-versa

`UL` Unknown Line, example GOTO 1000 when line 1000 doesn't exist.

Printed messages:

Those appear on the printer when the display can't be used:

`TransmissionError 10`
Use of the UC-2200 when the watch isn't in TRANSMIT mode.

`System Error 8`
Total hang of the system and reset.

The lacks

Some usual functions of the Microsoft BASIC are missing:

- PEEK/POKE and assembly access CALL/USR
- DATE$, TIME$, TIMER (why on a watch??)
- SAVE/LOAD system as there is a Backup for the Memo.
- INSTR to find a substring.
- XOR logical operator

Last Minute

Unexpected instructions were found, I didn't thought they were built in regarding the size of the text screen!

TAB(n)

Use with PRINT, specifies how many spaces before next printing:

PRINT TAB(3);A$

Note: if n>10 then the next line is used.

POS(0)

Pseudo-variable that returns the current cursor column from 0 to 9.

CSRLIN

Pseudo variable that returns the current cursor line from 0 to 3.

LOCATE n1,n2

Places the cursor at the given location:

n1 is the column from 0 to 9

n2 is the line from 0 to 3.

Chapter 4:
ROM Applications

Startup

- Mount the watch (UC-2000) on the Transmission Circuit
- Push the TRANSMIT button, you should see:
  - "TRANSMIT"
  - STAND-BY
- Switch the keyboard (UC-2200) ON and push the APL button, you should see:

```
1. DEMO
2. SCHEDULE
3. HIT
4. RACE
5. AMIDA
6. CARD
```

To select the line you want, use the **down and up arrows** and then **ENTER**.

DEMO

Nothing to do, just a looping demonstration of the system features.To stop and return to the main menu use the STOP key.

SCHEDULE

You're diven to a submenu:

```
1. INPUT
2. COPY
3. TRANSMIT
```

If the program is not yet loaded into the watch, you must start with Option 3 and transmit the data to the watch. See this option.

INPUT:

The date appears on the top line and you can note your appointments or else for that day in the last two lines.

F2 goes to the next day
Shift+F7 returns back one day

When you've done, press STOP or Shift+F10 to go back to the menu after a short transmission of the new data to the watch.

COPY:

Prints your schedule, you get something like that:

```
* COPY (Schedule) *

  8- 8 | Nothing
   FRI | to do
```

```
  8- 9 |
   SAT |

  8-10 | Train Bob
   SUN | 10h30 am

*****   END   *****
```

You can stop the printing holding down the STOP key.

TRANSMIT:

This initializes the program if it was not in memory before.

You are asked for the last two digits of the current year, and data is loaded into the watch. That's all!

Reading your schedule with the watch alone:

With the MODE button, you can see the schedule for the current day.

Use SET to go to the next day.

Use SELECT to go back one day.

Use MODE again to exit to the normal display.

HIT

It's a shoot'em up game for your watch.

Only one choice: `1) TRANSMIT`, so do it!
Then the game is playable with the watch only.

Playing the game:

With MODE go to `HIT GAME` screen and SELECT to play. You see that screen:

```
TIME   100
POINT    0
----------
 READY  ?
```

Use SELECT when you are ready.
During the game:
SELECT to move your shuttle (only one line down!)
SET to shoot the ennemies.

RACE

It's a game where you bet on four men races.

Only one choice: `1) TRANSMIT`, so do it!
Then the game is playable with the watch only.

Playing the game:

With MODE go to RACE screen and SELECT to play. You see that screen:

```
>1LANE:1.8
 2LANE:3.3
 3LANE:7.5
 4LANE:2.3
```

This screen shows the four lanes and their odds.

With SET move the ">" cursor in front of the one you want to bet on.

With SELECT validate your choice, you go to that screen:

```
 3LANE:7.5


$100   $00
```

With SET select the amount of money you want.

With SELECT validate your choice and start the race!

At the end of the race, the winner is displayed and you can see how much money you have. Press SELECT for another race.

AMIDA

It's a japanese game with four players (but it can be from 2 to many more) used to decide at random between the players. For example, we are four, in wich order will we wash the dishes this week? This program can help you…

Rules are very well explained at:
http://www.geocities.com/Athens/Acropolis/7247/amidakuji.html
If this link is dead, then just download the archive from my site, as the author made his explanations free:
http://pagesperso-orange.fr/gtello/amida.zip

Only one choice: `1) TRANSMIT`, so do it!
Then the game is playable with the watch only.

Playing the game:

With MODE go to `AMIDA KUJI` screen and SELECT to play. You see a screen with the upper part of the lines.

Each player makes his sign appear with SET and with the same button moves it to the place he wants.and fixes it with SELECT. You can put less than 4 signs.

When everyone is placed, push again SELECT to start the moving phase. Every sign gets down and reveals its number from 1 to 4, this deciding for you!

CARD

It's a Memory-like game where you must find pairs of signs. You play against the computer.

Only one choice: `1) TRANSMIT`, so do it!
Then the game is playable with the watch only.

Playing the game:

With MODE go to `30 CARDS` screen and SELECT to play. You see that screen:

```
LEVEL?
>LEVEL1
 LEVEL2
```

With SET move the ">" on the level you want.
With SELECT validate your choice. The game screen appears:

```
YOU- 0vs 0
□□□□□□□□□□
□□□□□□□□□□
□□□□□□□□□□
```

You see the back of the 30 cards.

Player turn: with SET move the cursor to the card you want to see and SELECT, repeat for the second card.

If both signs are equal, you win a point and can play again. If signs are not equal, then the computer plays.

At the end of the game, the scores and the winner are displayed.

Use SELECT to start a new game.

Chapter 5: Calculator

Startup

- Mount the watch (UC-2000) on the Transmission Circuit
- Push the TRANSMIT button, you should see:
  - "TRANSMIT"
  - STAND-BY
- Switch the keyboard (UC-2200) ON and push the Shift+Cal button, the screen turns to blank ready for calculations.
- To exit the calculator, press STOP.

Calculations

To enter the values use the 10 digits plus the "."

1 2 3 5 6 7 8 9 0

The other keys you can use are the one enclosed in a rectangle on the keyboard:

$+ \quad - \quad \times \quad \div \quad = \quad . \quad CE \quad AC$

Limits:

1) **No priority is known**, operations are performed in the order you type them, example:

$$12 + 3 \times 7 =$$

Gives the result 105.

2) The numbers are in **fixed point with 8 digits precision**, so the numbers are from 0,0000001 to 99999999.

Well, this is too bad knowing that the BASIC is present in ROM with scientific functions… The calculator could have been a little better…