Agilent 892X Family of Instruments
Key Programming Techniques

Product Note

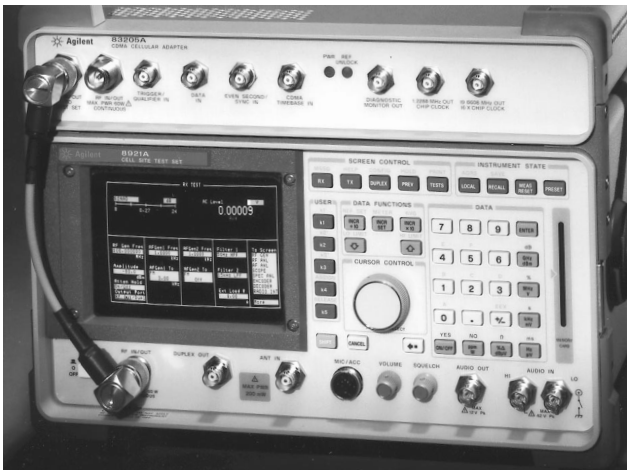

Table of Contents

Agilent Technologies

Innovating the HP Way

Introduction

This product note describes some key programming techniques for the Agilent Technologies 892X family of products. This document can be used as a training tool for users that are new to programming test equipment, or it can be used by experienced programmers that need to know about unique programming required for the 892X family of instruments.

The 892X family of products refers to Agilent 8920, 8921, 8922, and 8924. Most topics in this document apply to all of these products, however the examples on cellular call processing are specific to the 8920A/B and 8924C.

To simplify this document, the discussion and examples are written using BASIC on an external computer. For programmers using languages other than BASIC, the concepts and GPIB command syntax will still be important and helpful.

There are many documents from Agilent Technologies that are useful to learn about programming the 892X family of instruments. Some suggested documents are:

- *Agilent 8920/21 Programmers Guide,* part number 08920-90204
- *Manual and Automated IS-98 Testing with the 8924C* (Also useful for 8920A/B cellular programming), part number 5965-4775E
- *GPIB Instrument Programming Using Series 200/300 BASIC,* part number 50011-60032

Your comments and feedback on this document are important to Agilent Technologies. Please e-mail or FAX your suggestions to:

Jason Gallo
e-mail: jason_gallo@agilent.com
FAX 1-509-921-3700

Basic Input and Output from the Agilent 892X

Sometimes the most difficult part of controlling the test set is just getting started with your first measurement. Before attempting to automate a measurement, it is generally recommended to first perform the set up and make measurements manually, then duplicate the manual procedure with GPIB command strings and programming algorithms.

There are two fundamental processes for communicating with the 892X over the GPIB bus. The first process is sending data (like commands) to the instrument. In BASIC, the command to send data to the instrument is the OUTPUT statement. The second process for communicating with the 892X is reading data back from the instrument. In BASIC, the command to read data from the instrument is the ENTER statement.

Examples for using the OUTPUT statement:
An example two line program for sending a command to the 892X is:

```
10    OUTPUT 714;“*RST”
20    END
```

Explanation:
1. Line 10 sends a command to the instrument at address 714. The command is “*RST” which instructs the instrument to reset.

2. Line 20 is the END statement which is always required in any BASIC program. This line has no effect on the instrument.

To perform the initial set up for measurements on the 892X, the general process is:

- Use the OUTPUT statement to display the correct screen.
- Use the OUTPUT statement for making the settings on that screen.

An example of this would be:

```
10    OUTPUT 714;“DISP RFG”
20    OUTPUT 714;“RFG:FREQ 825 MHZ”
30    END
```

Explanation:
1. Line 10 displays the RF Generator Screen.

2. Line 20 sets the RF Generator frequency to 825 MHz.

This process is repeated over and over until all the necessary settings are made.

When the instrument settings are made and the radio under test is configured, the next step is to read a measurement from the instrument. In BASIC the ENTER statement is used to read data from an instrument. The ENTER statement is always preceded by an OUTPUT statement that tells the instrument which measurement the program wants to read.

Examples for using the ENTER statement:

```
10    OUTPUT 714;“MEAS:RFR:POW?”
20    ENTER 714;Data
30    END
```

Explanation:
1. Line 10, the OUTPUT statement is used to instruct the instrument to measure RF power. This line contains a question mark “?” and is called a query.

2. Line 20 is the ENTER statement that will actually read the result back from the 892X. The program will hang on line 20 until the test set responds with data. You must always follow a query with an ENTER statement to read the data back from the instrument. You should include some type of time out statement in your program to handle the situation when the test set cannot make the measurement.

The basic concept of reading measurements from the Agilent 892X is quite simple, however there are many factors that makes reading data one of the most difficult automated processes. When making measurements you need to consider measurement settling, measurement triggering, and error handling if the measurement doesn't complete. An automated procedure for setting up and making measurements should include the following steps:

1. Configure the 892X and radio.

2. Display and Activate the measurement on the 892X.

3. Wait until the test set and radio have "settled."

4. Trigger the measurement.

5. Output a query to the test set to tell it which result you want.

6. Enter the data back from the test set.

7. If the data doesn't come back from the test set, send a device clear to the test set so that the query is not still pending.

If any of these steps are not done properly you will receive "Query Error" messages on the 892X and it is possible to lockup the test set if errors are not handled correctly.

A more complete example that includes everything except error handling is the following:

```
10   OUTPUT 714;"TRIG:MODE:RETR SINGLE"
20   OUTPUT 714;"DISP RFG"
30   OUTPUT 714;"RFG:FREQ 825 MHZ"
40   OUTPUT 714;"RFG:AMPL -10 DBM"
50   OUTPUT 714;"DISP RFAN"
60   OUTPUT 714;"MEAS:RFR:POW:STATE ON"
70   WAIT 2
80   OUTPUT 714;"TRIG"
90   OUTPUT 714;"MEAS:RFR:POW?"
100  ENTER 714;Data
110  PRINT Data
120  END
```

Explanations:

1. Line 10 sets the instrument for single triggering.

2. Line 20 displays the RF Generator screen to allow settings.

3. Lines 30 and 40 set the RF Generator frequency and amplitude.

4. Line 50 displays the RF Analyzer screen.

5. Line 60 enables the RF power measurement. Once the measurement is enabled, it will remain active when ever a screen is displayed that contains the RF power measurement.

6. Line 70 is a wait statement that allows settling for the 892X and the radio under test.

7. Line 80 triggers the measurement.

8. Line 90 instructs the instrument to put the RF power result in the output queue to be read back by the controlling program.

9. Line 100 enters the data from the 892X output queue.

10. Line 110 prints the result to the computer screen.

More details on triggering and error handling are included later in this document.

Using Status Registers for Cellular Call Processing

Cellular call processing refers to registrations, originations, pages, and handoffs for cellular mobiles. This section applies to the 8920A/B and 8924C. These test sets contain built in call processing to perform all these cellular functions. Because executing these functions will take an unknown amount of time, it is necessary for the control program to use special techniques for monitoring the state of the test set and taking special care to keep the control program synchronized with the test set. This is accomplished by monitoring the status registers which indicate the current state of the test set. Each bit of the status registers corresponds to a unique state or process in the instrument. For a very detailed explanation of the capabilities, refer to the *Agilent 8920/21 Programmers Guide,* part number: 08920-90204.

The concept of writing programs that use status registers is quite simple, but since many users have never programmed the status registers it may seem a little difficult at first. The general procedure is to execute a call processing function (like maybe a PAGE), and then monitor one of the bits in the call processing status register group until the bit indicates that the function has completed. For instance, when executing a page, your control program can monitor the "connected" bit to indicate that the page was successful and the control program can continue.

In these programming examples the call processing *condition* register is monitored. This register indicates the current instantaneous state of the call processing system. In some programming situations it desirable to monitor the call processing event register. The *event* register latches changes in the condition register.

A brief programming example will help:

```
10   OUTPUT 714;"DISP ACNT"
20   OUTPUT 714;"*CLS"
30   OUTPUT 714;"CALLP:PAGE"
40   REPEAT
50      OUTPUT 714;"STAT:CALLP:COND?"
60      ENTER 714;Status_register
70      WAIT .2
80   UNTIL BIT(Status_register,5)
90   PRINT "Page was Successful!"
100  END
```

Explanations:

1. Line 10 displays the Analog Call Control screen. This is the screen where call processing functions take place.

2. Line 20 is a command to "Clear Status Registers". This is important because the program will monitor these registers to determine the instrument state.

3. Line 30 commands the 892X to PAGE the mobile.

4. Lines 40 and 80 create a loop to monitor the call processing status condition register.

5. Line 50 queries the 892X for the call processing status condition register.

6. Line 60 reads the value back from the test set and stores the data in a variable called Status_register.

7. Line 70 is a short wait (200 milliseconds) which allows the test set some time to process other tasks. This is important because the GPIB queries have a high priority and if the loop doesn't have any delays, the test set is so busy reporting to the external computer that the other processes (like the page) may not complete.

8. Line 80 examines bit 5 of the variable Status_register. When bit 5 is set, it indicates the mobile is connected and only then should the control program continue on and make measurements.

This short routine to page the mobile is complete except for any error trapping. If the mobile doesn't connect, the program will stay in a continuous loop. There are two techniques that are commonly used for this situation.

The first technique involves monitoring the "Event Status Register," which is where error conditions are reported. The second technique involves putting a timer into the loop and if the call processing function doesn't complete in a specified time, the control program can exit the loop and continue with some type of error trapping routine.

The following is an example of the technique that monitors the Event Status Register for errors:

```
10   OUTPUT 714;"DISP ACNT"
20   OUTPUT 714;"*CLS"
30   OUTPUT 714;"CALLP:PAGE"
40   REPEAT
50      OUTPUT 714;"STAT:CALLP:COND?"
60      ENTER 714;Status_register
70      OUTPUT 714;"*ESR?"
80      ENTER 714;Error_register
90      IF Error_register<>0 THEN
100     PRINT "Error during page"
110     STOP
120     END IF
130  WAIT .2
140  UNTIL BIT(Status_register,5)
150  PRINT "Page was Successful!"
160  END
```

Explanations:

1. Lines 70 and 80 are used to read the standard event status register.

2. Lines 90 checks if the variable Error_register is not equal to zero, which would indicate an error.

3. Lines 100-120 prints an error message and stops the program.

The following is an example of the technique that uses a loop counter as a timer to detect the page failure:

```
10   OUTPUT 714;"DISP ACNT"
20   OUTPUT 714;"*CLS"
30   OUTPUT 714;"CALLP:PAGE"
40   Loop_count=0
50   REPEAT
60      Loop_count=Loop_count+1
70      OUTPUT 714;"STAT:CALLP:COND?"
80      ENTER 714;Status_register
90      IF Loop_count=50 THEN
100     PRINT "Error during page"
110     STOP
120     END IF
130     WAIT .2
140  UNTIL BIT(Status_register,5)
150  PRINT "Page was Successful!"
160  END
```

The process to use status registers for performing registrations, pages, originations, and handoffs are all similar. The following example shows how to perform a registration. One small difference you may notice in this example is the monitoring loop is looking for the Agilent 892X to leave the registration condition (indicating the registration is finished). In the example program for a page, the monitoring loop was waiting for the 892X to enter the connected condition (indicating that the page state was finished).

```
10    OUTPUT 714;“DISP ACNT”
20    OUTPUT 714;“*CLS”
30    OUTPUT 714;“CALLP:REGISTER”
40    WAIT 2
50    Loop_count=0
60    REPEAT
70       Loop_count=Loop_count+1
80       OUTPUT 714;“STAT:CALLP:COND?”
90       ENTER 714;Status_register
100      IF Loop_count=50 THEN
110      PRINT “Error during registration”
120      STOP
130      END IF
140      WAIT .2
150   UNTIL NOT BIT(Status_register,1)
160   PRINT “Registration was Successful!”
170   END
```

Explanations:

1. Line 10 displays the Analog Call Control screen. This is the screen where call processing functions take place.

2. Line 20 is a command to “Clear Status Registers”. This is important because the program will monitor these registers to determine the instrument state.

3. Line 30 commands the 892X to REGISTER the mobile.

4. Line 40 is a two second wait that allows the test set to enter the “registration” state.

5. Lines 50 to 150 create a loop to monitor the call processing status condition register.

6. Line 70 is a loop counter to know when to exit because of a time out.

7. Line 80 and 90 queries the 892X for the call processing status condition register.

8. Lines 100-130 checks the loop counter and stops the program if the count = 50.

9. Line 140 is a short wait (200 milliseconds) which allows the test set some time to process other tasks.

10. Line 150 monitors the “registration bit” until the 892X leaves the registration state. This is indicated by the bit being set to zero (false).

Triggering Measurements

There are two types of triggering modes for remote (automated) operation. When the 892X is in remote mode, the trigger is set to either repetitive (continuous) or single. In repetitive triggering mode, all active measurements are continuously triggered. When the control program queries the 892X for a measurement result, the 892X triggers a new measurement and places the last previously completed measurement result in the output queue for the control program to read.

When using single trigger, the control program must send a “TRIG” command to the 892X. This flushes all the current measurement results and causes the 892X to begin a new measurement. The new measurement will update the results for all the active measurements. After the measurement results are updated, the control program can query the 892X for results. This causes the 892X to place the requested results in the output queue where the control program can then retrieve the measurement results.

Two short examples will demonstrate the two different triggering modes:

Repetitive Triggering example:

```
10   OUTPUT 714;”TRIG:MODE:RETR REPETITIVE”
20   OUTPUT 714;“DISP RFAN”
30   OUTPUT 714;“MEAS:RFR:POW?”
40   ENTER 714;Result
50   PRINT Result
60   END
```

Explanation:

1. Line 10 sets the remote triggering mode to repetitive (continuous).

2. Line 20 displays the RF Analyzer screen. This will automatically start triggering for all the active measurements on that screen.

3. Line 30 queries the test set for the RF Power measurement. This causes the last result to be put into the output queue.

4. Line 40 enters the data from the output queue into the variable called “Result”.

Single Triggering example:

```
10   OUTPUT 714;“TRIG:MODE:RETR SINGLE”
20   OUTPUT 714;“DISP RFAN”
30   OUTPUT 714;“TRIG”
40   OUTPUT 714;“MEAS:RFR:POW?”
50   ENTER 714;Result
60   PRINT Result
70   END
```

Explanation:

1. Line 10 sets the remote triggering mode to single.

2. Line 20 displays the RF Analyzer screen. No measurements will appear.

3. Line 30 triggers all the active measurements on the RF Analyzer screen.

4. Line 40 queries the test set for the RF Power measurement. This causes the result from the previous trigger to be put into the output queue.

5. Line 50 enters the data from the output queue into the variable called “Result”.

Handling GPIB Query Errors

Under certain conditions, it may not be possible for the Agilent 892X to make a measurement. If this happens, the control program must properly clear the GPIB buffer before proceeding with any other program statements. A common mistake is when a measurement doesn't complete, the control program is stopped and rerun. Quite often the first command sent to the instrument is "*RST" for an instrument reset. This is not the correct way to handle the query error and it can lockup the test set under certain conditions. The correct technique would be to send a "device clear" to the instrument before any commands are sent. It is a good idea to have the very first statement in any GPIB control program to be a device clear.

The device clear (or selected device clear) command is defined in IEEE 488 and it instructs the instrument to clear any pending GPIB operations. The device clear statement is unique for each different interface card and programming language. As an example, in BASIC the command "CLEAR 714" sends a selected device clear to the instrument at address 714. In a language like C or Pascal, the command could be "IOCLEAR (714)" or "IOCLEAR (714L)". Refer to the documentation for your programming language and GPIB interface card.

An example of a good programming technique to start your control program could be:

```
10   CLEAR 714
20   OUTPUT 714;"*RST"
30   END
```

By starting the program with a device clear, any pending GPIB operations will be cleared and there won't be any chance of locking up the 892X. Typically the control program will have some type of error trapping to clean up the GPIB queue in case of measurement errors or measurements that don't complete. A simple example can illustrate how a programmer could implement this in BASIC.

```
10   Start_meas: !
20   ON TIMEOUT 7,15 GOTO Clean_up_gpib
30   OUTPUT 714;"TRIG:MODE:RETR SINGLE"
40   OUTPUT 714;"DISP RFAN"
50   OUTPUT 714;"TRIG"
60   OUTPUT 714;"MEAS:RFR:POW?"
70   ENTER 714;Result
80   PRINT Result
90   STOP
100  Clean_up_gpib:!
110  CLEAR 714
120  PRINT "Measurement was aborted"
130  GOTO Start_meas
140  END
```

Explanations:

Line 20 is an BASIC statement that sets up a time-out. This statement instructs BASIC to go to the line labeled "Clean_up_gpib" if any instrument on the GPIB bus 7 doesn't respond for 15 seconds. (In this example, 7 is the interface select code). If the power measurement doesn't complete, the program will hang up on line 70 for 15 seconds while it is waiting for data back from the 892X. After the 15 second "ON TIMEOUT" timer expires, the program jumps to line 100, then executes line 110 to clear the GPIB operations. Line 120 prints an error message and line 130 causes the program to go to the beginning (Line 10) and start over.

This is a simple technique that can be implemented in almost any programming language. It is a good programming practice that whenever data is to be sent to the instrument or read from the instrument, some type of error handling routine should be used. One implementation would be to write general purpose subroutines like "To_testset" or "From_testset" that would send data back and forth between the computer and the test set. These general purpose routines would be a good location in the code to include the proper error handling routines.

Reducing Test Time

Writing the fastest possible test software involves efficient programming, understanding the device-under-test, and optimizing set ups and measurements with the test equipment. This section of this document focuses on the test set, and particularly on helping the programmer understand and avoid some of the slower (and often unnecessary) processing in the test set.

The following processes in the 892X family are relatively slow compared to automated applications.

- Changing screens when it isn't necessary.
- Auto-ranging and auto-tuning.
- Making unnecessary measurements.

Changing screens on the 892X takes approximately one second. For making *measurements* it is generally necessary to be on the correct screen, however many *settings* (such as the RF Generator) can be made from almost any screen. For analog measurements, try to use the DUPLEX screen because it contains Generator and Analyzer functions. Avoid the Analog RX and TX screens because they auto-configure many settings in the test set and this is relatively slow.

Auto-ranging and auto-tuning for the analyzer are two processes that should definitely be avoided. Auto-ranging is the process where the test set analyzer automatically sets the correct gain and attenuation setting for a measurement. Auto-tuning in the test set is where the RF analyzer will automatically find, and tune to, the largest RF signal. If you know the frequency and level of the signal-under-test, set the 892X tuning mode to "manual" and set the input attenuators to "hold". Write your program to directly control the analyzer frequency and input attenuation. To find the correct attenuation, you may want to manually allow the 892X to auto-range on a particular signal, then in your automated application you can set the gain correctly under program control.

It is obvious that making unnecessary measurements will take additional time, however many programmers don't realize that their application software is often causing the test set to make unnecessary measurements. There are two situations that cause unnecessary measurements. The first situation is when the displayed screen has more measurements activated than necessary. For instance, the RF Analyzer screen may have Frequency Error, TX Power, FM Deviation, and AF Frequency all active. Whenever a trigger command is issued, the test set will automatically make all the active measurements. You can save a significant amount of time by disabling the unused measurements. A sample of how to program the 892X to disable the FM Deviation measurement is:

```
OUTPUT 714;"MEAS:AFR:FM:STATE OFF"
```

A second situation that often generates unnecessary measurements is during repetitive triggering. When making multiple measurements with a single setting, use single triggering, and read back the results from all the measurements. This technique does not cause the 892X to re-trigger measurements each time a new query is sent to the test set.

For example:

```
10   OUTPUT 714;"TRIG:MODE:RETR SINGLE"
20   OUTPUT 714;"TRIG"
30   OUTPUT 714;"MEAS:RFR:POW?"
40   ENTER 714;Pow_result
50   OUTPUT 714;"MEAS:AFR:FM?"
60   ENTER 714;Fm_result
70   PRINT Pow_result,Fm_result
80   END
```

In this example, line 20 triggers all the active measurements, and lines 30 to 60 read back results.

Another example of essentially the same program is:

```
10    OUTPUT 714;“TRIG:MODE:RETR SINGLE”
20    OUTPUT 714;“TRIG”
30    OUTPUT 714;“MEAS:RFR:POW?;:MEAS:AFR:
      FM?”
40    ENTER 714;Pow_result,Fm_result
50    PRINT Pow_result,Fm_result
60    END
```

This program saves some GPIB traffic and reads back multiple measurements by combining two queries and two Enters on the same line. (This technique may be more difficult to implement in languages other than BASIC).

Agilent Technologies' Test and Measurement Support, Services, and Assistance

Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise

"Our Promise" means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

Your Advantage

"Your Advantage" means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

By internet, phone, or fax, get assistance with all your test and measurement needs.

Online Assistance

www.agilent.com/find/assist

Phone or Fax

United States:
(tel) 1 800 452 4844

Canada:
(tel) 1 877 894 4414
(fax) (905) 206 4120

Europe:
(tel) (31 20) 547 2323
(fax) (31 20) 547 2390

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Latin America:
(tel) (305) 269 7500
(fax) (305) 269 7599

Australia:
(tel) 1 800 629 485
(fax) (61 3) 9210 5947

New Zealand:
(tel) 0 800 738 378
(fax) (64 4) 495 8950

Asia Pacific:
(tel) (852) 3197 7777
(fax) (852) 2506 9284

Product specifications and descriptions in this document subject to change without notice.

Printed in U.S.A. 9/00
5965-6120E

For more information, visit our website at
www.agilent.com/find/8920support/

Agilent Technologies
Innovating the HP Way