2920 SOFTWARE SUPPORT PACKAGE

- **Complete software design and development support for the 2920**
- **Extends Intellec® Microcomputer Development System to support 2920 software development**

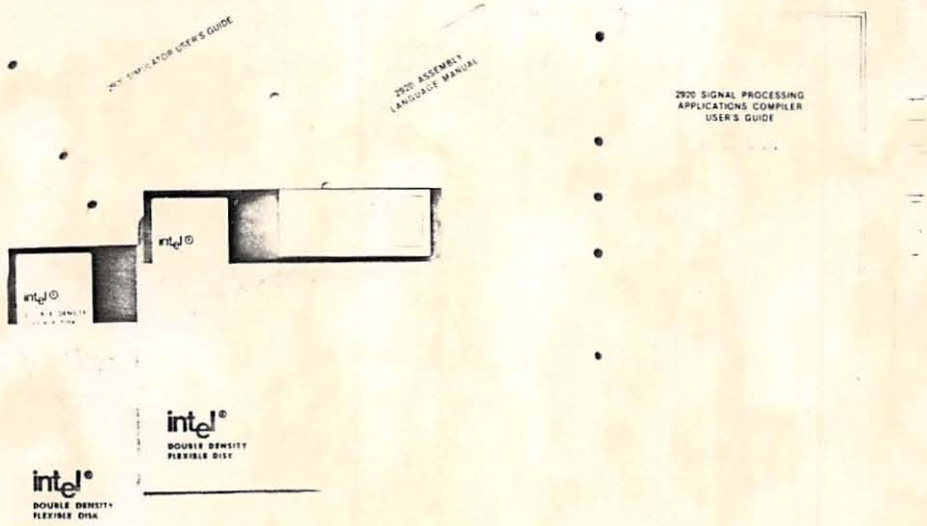
The 2920 Software Support Package furnishes a 2920 Signal Processing Application Compiler, 2920 Assembler, and 2920 Software Simulator. These three software design and development tools run on the Intellec® Microcomputer Development System.

The 2920 Signal Processing Application Compiler is an interactive tool for designing software to be executed on the 2920 Signal Processor. The compiler accepts English-like statements from the user and generates 2920 assembly language code.

The assembler translates symbolic 2920 assembly language programs into the machine operation codes. The user can load the codes into the simulator for 2920 simulation or to the Universal PROM Programmer for 2920 EPROM programming.

The simulator, operating entirely in software, allows the user to test and symbolically debug 2920 programs. The user can specify input signals, simulate program execution, set up breakpoints, display input and output, and display and alter the contents of the 2920 registers and memory locations. The simulator can also stop or trace the program and constructively give the user access to the key elements inside a 2920 for analyzing his program.

The compiler, assembler, and simulator enable the designer to develop and test an entire program without a complete prototype design. The 2920 designer works on the Intellec® Microcomputer Development System rather than on a breadboard. The development system can program, store and recall programs or routines and aid in 2920 program design.



2920 Software Support Package

May 1980
1662208

2920 SIGNAL PROCESSING APPLICATIONS COMPILER

- **Compiler generates 2920 Assembly Language Code**
- **Extensive command set for designing electrical filters**
- **Graphics capability enhances analysis of filter response**
- **Powerful MACRO capability for executing frequently used routines**
- **Interactive software support tool for 2920 Signal Processor**
- **Extends Intellec® Microcomputer Development System support of the 2920**
- **Contains MACRO library for several standard filters and signal processing functions**

The 2920 Signal Processing Applications Compiler (SPAS20) is an interactive tool for designing software to execute on the 2920 Signal Processor.

The SPAS20 package can be visualized as being comprised of three inter-related sections: A compiler section, a filter design section, and a MACRO section.

Among the capabilities of SPAS20 are: ability to generate 2920 assembly language code directly from specifications of signal processing building blocks such as filters and waveform genrators; ability to generate 2920 assembly language code for several classes of algebraic equations such as Y=C * X, Y=C * Y, and Y=C * X + Y where X,Y are variables and C is a constant; ability to examine time and frequency responses of filter sections specified by continuous or sampled poles and zeroes; ability for users to implement more complex commands by grouping sets of commonly used commands into a MACRO.

The SPAS20 package runs under ISIS-II on any Intellec Microcomputer Development System with 64K RAM. The output of SPAS20 can be assembled with the 2920 assembler, tested with the 2920 Simulator, and programmed into the 2920 chip with the Universal PROM Programmer for prototyping.

FUNCTIONAL DESCRIPTION

The 2920 Signal Processing Applications Compiler gives the analog designer a "high level language" for his 2920 applications—it decreases the need to code 2920 assembly language. Furthermore, the compiler is interactive. This feature enables the designer to define a filter, graph its response, and change its parameters many times, without having to program and test the filter in an actual 2920 implementation. The command language is very similar to that of Intel's In-Circuit Emulators.

Once a filter is realized by moving poles and zeros in the continuous and sampled planes, the filter may be coded and written onto an ISIS file. Several other file commands are available to store and retrieve command sequences for SPAS20 sessions.

SPAS20 Command Language

DEFINE	This command defines a pole or zero by associating it with a number (i.e., POLE 3), and with real and imaginary coordinates in the continuous or sampled plane. This command also defines a symbol by associating a name with a numeric value, or a MACRO by providing a pointer to a specified command sequence.
GRAPH/ OGRAPH	This command graphically displays the values of object(s) specified. For example, GRAPH GAIN and GRAPH PHASE are used to display filter response. The OGRAPH command will "overgraph" the new response over the old response, after any changes have been made. (You may also graph Group Delay, Step, and Impulse.)
MOVE	Allows the definition of a pole or zero to be changed—its coordinates, its plane, or both.
REMOVE	Deletes the definition of a pole, zero, symbol, or macro.
HELP	Types an explanatory message on the console, pertaining to a command or its attributes.
HOLD	Command to correct attenuation due to sample-and-hold distortion: if ON, it corrects absolute gain by sin(x)/x and phase by adding x, where x=TS*FREQ*π. It corrects group delay by subtracting π*TS.
EVALUATE	Gives the decimal numeric value of any expression.
CODE	Creates 2920 assembly language code for given poles, zeros, and equations.

The SPAS20 compiler also recognizes the following commands for file handling:

PUT/ APPEND	Writes out objects (commands) to a specified file, either creating a new one or appending an existing one. This enables the user to store all or part of a SPAS20 session on a diskette to be brought back later with the INCLUDE command.
DISPLAY	Copies the contents of a file to the console.
INCLUDE	Executes a sequence of instructions from a diskette file as if they were typed in from the console.
LIST	Creates a file containing all console interactions.

In addition to naming macros for specific command sequences, compound and conditional commands may be formed using all of the above statements. These compound commands are:

IF	Establishes conditional flow of control within a block of commands.
REPEAT	Used for repetition of a block of commands; executes indefinitely or until a condition is met (using WHILE, UNTIL, and END statements).
COUNT	Establishes the number of times a command sequence is to be executed, in a looping fashion.

SPAS20 MACRO Facility

A macro is a sequence of commands that is stored on a temporary diskette file. The command sequence is executed when the macro name is entered as a command. This saves repetitive entry of the sequence, and permits alogorithms to be saved on diskette for future use. This SPAS20 facility allows you to do the following:

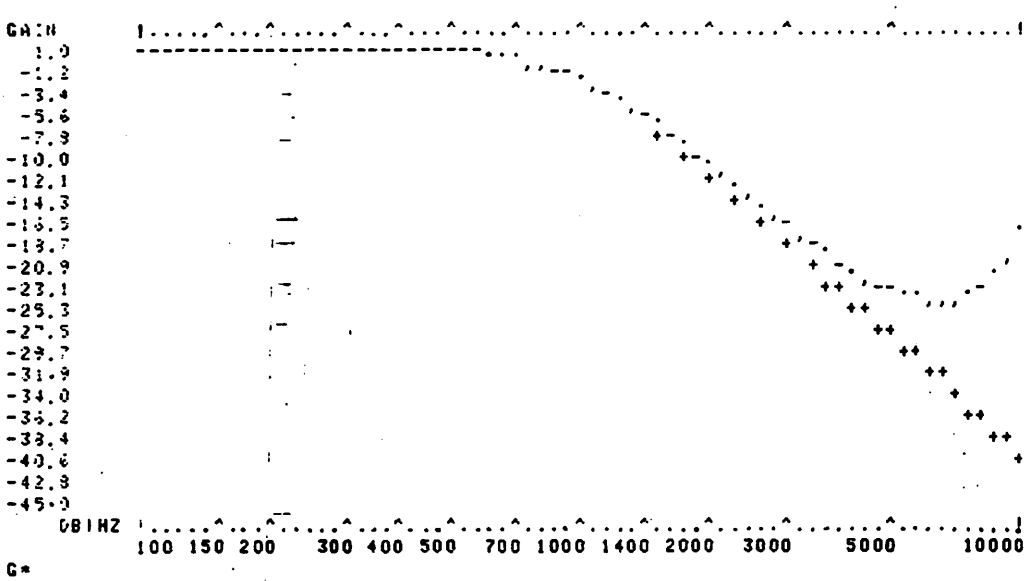
- Display the text of any macro.
- Define a macro, specifying its name and any parameters that are to be used by the block. This definition is followed by the contents of the macro (commands) and the EM statement to end its definition.
- Invoke a macro by entering its name and appropriate values for any parameters.
- List the names of all defined macros.
- Remove any or all macros.

SAMPLE SPAS20 SESSION

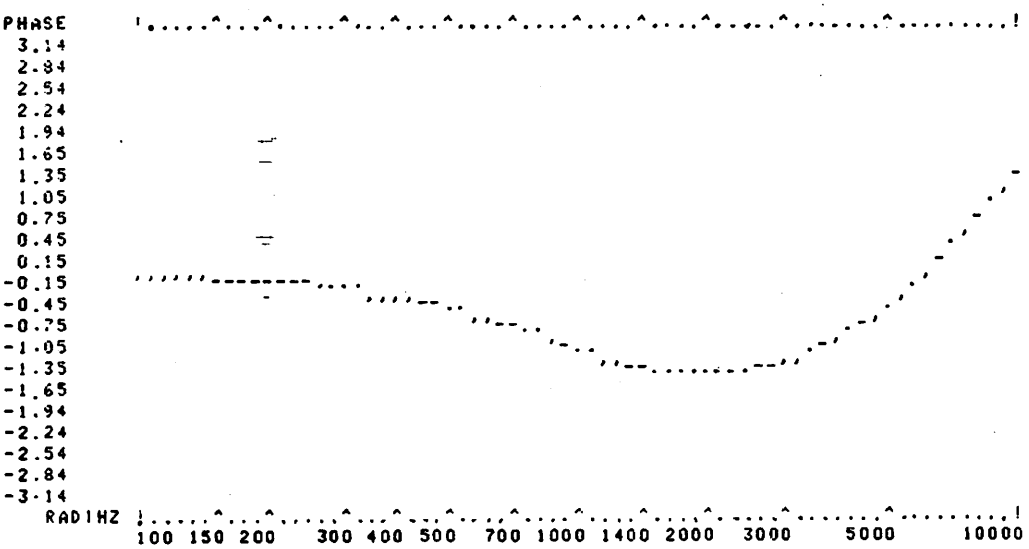
```
-:F1:SPAC20.SFT

ISIS-II 2920 SIGNAL PROCESSING APPLICATIONS COMPILER, V1.0
*
*DEFINE POLE 1 = -707,707        ; CREATE A POLE IN CONTINUOUS S-PLANE
*
*PZ        ; LIST ALL POLES AND ZEROS
POLE 1 = -707.00000,707.00000,CONTINUOUS
*
*FSCALE = 100,10000        ; ESTABLISHES FREQUENCY RANGE OF INTEREST
*
*YSCALE = -45,1        ; ESTABLISHES MAGNITUDE RESPONSE RANGE OF INTEREST
*
*GRAPH GAIN        ; PLOT MAGNITUDE RESPONSE OF POLE PAIR

GAIN      !.....^...^.....^...^...^....^....^....^....^.....^.......^.........!
  1.0     ---------------------------....
 -1.2                                   ''--.
 -3.4                                        '-.
 -5.6                                           '-.
 -7.8                                              '-.
-10.0                                                 '-.
-12.1                                                    '-.
-14.3                                                       '-.
-16.5                                                          '-.
-18.7                                                             '-.
-20.9                                                                '-.
-23.1                                                                   '-.
-25.3                                                                      '-.
-27.5                                                                         '-.
-29.7                                                                            '-.
-31.9                                                                               '-.
-34.0                                                                                  '-.
-36.2                                                                                     '-.
-38.4                                                                                        '-.
-40.6                                                                                           '-.
-42.8                                                                                              '-.
-45.0
    DB|HZ !.....^...^.....^...^...^....^....^....^....^.....^.......^.........!
          100 150 200   300 400 500  700 1000 1400 2000  3000    5000     10000
**
*
*; THE UNITS USED IN GRAPHING GAIN ARE SHOWN IN THE LOWER LEFT CORNER.
*; GAIN IN DECIBELS IS GRAPHED VERSES FREQUENCY IN HERTZ
*
*; PREPARE TO MOVE TO THE DIGITAL DOMAIN.
*; SAMPLE RATE MUST BE SPECIFIED.
*
*TS = 1/13020        ; RATE FOR 192 INSTRUCTION PROGRAM AND 10MHZ CLOCK
TS = 7.6805004/10**5
*
```

SAMPLE SPAS20 SESSION (Cont'd.)

```
*MOVE POLE TO Z        ; CONVERT FILTER TO DIGITAL VIA MATCHED-Z TRANSFORMATION
1 POLES/ZEROES MOVED
*
*P        ; LIST TRANSFORMED POLE
POLE 1 = 0.71092836,0.34118369,Z
*
*; COMPARE RESPONSES OF THE ANALOG AND DIGITAL FILTERS BY GRAPHING THE
*; NEW RESPONSE OVER THE OLD.
*
*OGRAPH GAIN
```




```
G*
*
*; PLUS SIGNS INDICATE OLD CURVE.
*; NOTE THAT THE DIGITAL FILTER RESPONSE BEGINS TO INCREASE AGAIN
*; AT HALF THE SAMPLE RATE ( 6510 HZ ).
*
*; THE PHASE CHARACTERISTICS OF THIS FILTER CAN BE EXAMINED.
*
*YSCALE = -PI,PI        ; ESTABLISHES RANGE OF INTEREST
*
*GRAPH PHASE
```




```
P*
*
*PUT :F1:POLE PZ        ; SAVE THE POLE LOCATION IN A DISK FILE BACKUP
*
*CODE POLE 1 INST(11        ; GENERATE 2920 ASSEMBLY CODE FOR THIS FILTER
B:=1 33989990  B2=-0.50541914
```

SAMPLE SPAS20 SESSION (Cont'd.)

```
      ( OPTIMIZED 2920 CODE IS NOW GENERATED. TO SAVE SPACE, SOME
      OF THE SCREEN OUTPUT HAS BEEN DELETED. NORMALLY ALL ATTEMPTS
      BY THE COMPILER TO GENERATE CODE ARE ECHOED ON THE SCREEN. )

INST=10
 POLE 1 = 0.71089458,0.34116779,Z
  BEST: PERROR = 3.3795874/10**5,1.58846567/10**5

; NOTE: MAKE SURE SIGNAL IS <0.74635571
LDA OUT2_P1,OUT1_P1,R00
     ; OUT2_P1=1.00000000*OUT1_P1
LDA OUT1_P1,OUT0_P1,R00
     : OUT1_P1=1.00000000*OUT0_P1
SUB OUT0_P1,OUT1_P1,R05
     ; OUT0_P1=1.00000000*OUT0_P1-0.031250000*OUT1_P1
ADD OUT0_P1,OUT0_P1,R03
     : OUT0_P1=1.12500000*OUT0_P1-0.035156250*OUT1_P1
ADD OUT0_P1,OUT1_P1,R02
     ; OUT0_P1=1.12500000*OUT0_P1+0.21484375*OUT1_P1
SUB OUT0_P1,OUT2_P1,R01
     ; OUT0_P1=1.12500000*OUT0_P1+0.21484375*OUT1_P1-0.50000000*OUT2_P1
SUB OUT0_P1,OUT2_P1,R08
     : OUT0_P1=1.12500000*OUT0_P1+0.21484375*OUT1_P1-0.50390625*OUT2_P1
ADD OUT0_P1,OUT2_P1,R11
     : OUT0_P1=1.12500000*OUT0_P1+0.21484375*OUT1_P1-0.50341796*OUT2_P1
SUB OUT0_P1,OUT2_P1,R09
     : OUT0_P1=1.12500000*OUT0_P1+0.21484375*OUT1_P1-0.50537109*OUT2_P1
ADD OUT0_P1,IN0_P1,R00
     ; OUT0_P1=1.12500000*OUT0_P1+0.21484375*OUT1_P1-0.50537109*OUT2_P1+1.00000000*IN0_P1
*
*; THE CODE COMMAND SPECIFIED THAT THE POLE PAIR BE CODED IN LESS THAN 11
*; INSTRUCTIONS, SO 10 INSTRUCTIONS WERE GENERATED, WITH COMMENTS.
*; THE FINAL ERROR IN RADIUS AND ANGLE FOR THE POLE PAIR WAS OF THE
*; ORDER OF 1/10**5 AS INDICATED ABOVE IN PERROR.
*; THIS OPTIMIZED 2920 ASSEMBLY CODE CAN NOW BE APPENDED TO A FILE
*; WHICH MAY CONTAIN OTHER CODED FUNCTIONAL BLOCKS OF A 2920 PROGRAM
*
*EXIT
```

2920 ASSEMBLER

2920 program development on Intellec® Microcomputer Development Systems

Translates symbolic assembly language instructions into 2920 machine code

Produces Assembly Listing, Object Code File, and Error Diagnostics

Output used for 2920 programming with the Intellec PROM Programmer or the 2920 Simulator for program debug

The 2920 Assembler translates symbolic 2920 Assembly Language instructions into the appropriate machine operation codes. Through this facility, the programmer is able to symbolically program 2920 hardware operations. Compared to machine code, these symbolic references provide faster programming, easier debugging, and greater reliability.

The Assembler produces an object code file (executable machine code), a complete assembly listing, and error diagnostics. The object code output from the Assembler may be loaded directly into the Intel Universal PROM Programmer for programming the 2920 EPROM. The object code may also be loaded to the 2920 Simulator for 2920 system design and debug.

The 2920 Assembler runs under the ISIS-II Operating System on the Intellec Microcomputer Development Systems.

Sample 2920 Assembly Listing

```
ISIS-II 2920 ASSEMBLER X102                                    PAGE    1

ASSEMBLER INVOKED BY: AS2920 SAW.ASM DEBUG

SAWTOOTH WAVE GENERATOR

  LINE  LOC OBJECT SOURCE STATEMENT

     1              $TITLE('SAWTOOTH WAVE GENERATOR')
     2              ;
     3              ;
     4     0 0000EF     INO              ; SAMPLE INPUT CHANNEL 0
     5     1 0000EF     INO
     6     2 0000EF     INO
     7     3 008AEB     SUB Y,KP1,INO    ; SIMULTANEOUSLY CALCULATE SAWTOOTH
     8     4 008A0A     SUB Y,KP1,R1,INO ; BY SUBTRACTING 3/16 FROM Y
     9     5 0044EF     LDA DAR,Y,INO    ; ALSO CHECK SIGN BIT OF Y
    10     6 7A8AED     ADD Y,KP7,CNDS   ; IF Y NEGATIVE START NEXT TOOTH
    11     7 6000EF     CVTS             ; CONVERT SAMPLED INPUT TO DIGITAL (SIGN BIT)
    12     8 7082EF     LDA Y,KP0,CNDS   ; SUPPRESS SAWTOOTH IF INPUT WAS < 0
    13     9 4044EF     LDA DAR,Y        ; PREPARE TO OUTPUT SAWTOOTH
    14    10 4000EF     NOP              ; ANALOG LEVEL MUST SETTLE
    15    11 4000EF     NOP
    16    12 4000EF     NOP
    17    13 8000EF     OUTO             ; OUTPUT SAWTOOTH
    18    14 8000EF     OUTO
    19    15 8000EF     OUTO
    20    16 5000EF     EOP              ; PROGRAM WILL END IN THREE MORE INSTRUCTIONS
    21    17 8000EF     OUTO
    22    18 8000EF     OUTO
    23    19 8000EF     OUTO
    24              ;
    25                  END


SYMBOL:                         VALUE:

Y                                   0


ASSEMBLY COMPLETE
ERRORS     =     0
WARNINGS   =     0
RAMSIZE    =     1
ROMSIZE    =    20
```

2920 SIMULATOR

Speeds test and debug of 2920 programs

Simulates 2920 internal operation

Operates on Intellec® Microcomputer Development Systems

Allows users to specify 2920 input signals, and display or alter ROM, RAM, and system variables

Output and internal data can be saved on disk for further analysis.

Provides ability to set breakpoints and to collect trace information

Easy-to-learn commands

The 2920 Simulator is a software facility that provides testing and symbolic debugging of 2920 programs in an Intellec Microcomputer Development Systems environment. The 2920 designers have the capability to specify the 2920 input signals, to set breakpoints, to collect and display 2920 input, output, system variables, and ROM and RAM data values during simulation. The 2920 Simulator accepts the hex format object files produced by the 2920 assembler. Output values and internal trace data may be saved on ISIS-II disk files for further analysis.

Functional Description

2920 Input Signal Specification

The four analog signal inputs to the 2920 processor can be specified as algebraic combinations of basic functions of time. The basic functions are SIN, COS, EXP, LOG, SQR, SAW, SQW, ABS.

2920 Simulation

The simulation of 2920 machine instructions is performed in software. All 2920 internal registers, memory, input values, output values, and other system variables can be examined and modified. The internal processing of the 2920 is simulated. Time constants for the sample and hold capacitators are assumed to be zero. Calculation of input signals is performed in single precision floating point. The speed of simulation varies with the complexity of the input signal, breakpoint setting, and trace condition. Exclusive of I/O time requirements, 2920 instructions will be simulated at a rate of approximately several hundred instructions per second.

Breakpoint Capabilities

After each instruction is simulated, the breakpoint is evaluated to determine whether to stop or continue simulation. Conditional breakpoints are also provided for debugging purposes. Simulation can be manually stopped at any time by pressing the ESC key on the Intellec console.

Trace Capabilities

Based on the qualifier's condition, trace data records can be collected during simulation. The trace data records are stored in Intellec resident memory and are optionally written to the console for display or to a disk file for record.

Symbolic Debugging Capabilities

The 2920 Simulator allows the user to refer to program addresses symbolically. The user can load or save the symbols generated from the hex format object files or created during the debugging session. 2920 program memory in ROM can be disassembled, or filled with assembled instructions.

The 2920 Simulator is designed to provide users with powerful, easy-to-use commands. The user interfaces to the Simulator by entering commands to the Intellec console. The commands consist of one command line, terminated by one of the two line terminators — carriage return or line feed.

The 2920 Simulator offers two types of commands:

Simulation and Control Commands

Command	Operation
Simulate	Starts simulation of the input signals and the 2920 program in simulated ROM memory. Initial setting is "FOREVER."
Trace	Controls the trace selection. Initial setting is "TIME."
Qualifier	Sets qualifier condition during trace. Initial setting is "ALWAYS."
Breakpoint	Sets breakpoint condition during simulation. Initial setting is "NEVER."

Interrogation and Utility Commands

Command	Operation
Display	Displays the values of symbols, RAM, ROM, input, output, registers and system variables.
Change	Alters the values of symbols, RAM, ROM, input, register and system variables.
Base	Establishes the mode of display for output data.
Suffix	Establishes the mode of display for input data.
Load	Fetches user symbol table and object code from input device.
Save	Sends user symbol table and object code to output device.
Define	Enters symbol name and value to user symbol table.
Console	Controls the console on/off display.
List	Defines list device.
Exit	Returns program control to ISIS-II.
Evaluate	Converts expression to equivalent values in binary, decimal, and hex.
Remove	Deletes symbols from symbol table.
Help	Provides a brief summary of the syntax for the command languages.

Keyword References

The 2920 Simulator provides users with keyword references to gain access to all of the numeric valued system variables including simulated 2920's memory, register, status flags and input/output. These keyword references can function as the evaluation command, display command, and change command.

- **2920 Processor Keyword References**

IN0	Analog input 0 in volts
IN1	Analog input 1 in volts
IN2	Analog input 2 in volts
IN3	Analog input 3 in volts
OUT0	Analog output 0 in volts (read only)
OUT1	Analog output 1 in volts (read only)
OUT2	Analog output 2 in volts (read only)
OUT3	Analog output 3 in volts (read only)
OUT4	Analog output 4 in volts (read only)
OUT5	Analog output 5 in volts (read only)
OUT6	Analog output 6 in volts (read only)
OUT7	Analog output 7 in volts (read only)
IN	Sampled and held analog input signal in volts
DAR	Digital to analog register (RAM location 40)
PC	Program counter (integer 1 to 192)
CY	Carry (integer 0 or 1)
OVF	Overflow (integer 0 or 1, read only)
OVE	Overflow enable (integer 0 or 1)

- **Software Simulator Keyword References**

TIME	Elapsed simulated time in seconds (read only)
TQUAL	Time when the qualifier last matched in seconds (read only)
COUNT	Number of instructions simulated since last SIMULATE command (integer, read only)
BUFFERSIZE	Number of trace data records (integer, read only)
TINST	Time between successive instructions in seconds (read only)
SIZE	Number of instructions in program disregarding actual EOP placement
TPROG	Time between successive program passes in seconds
VREF	Reference analog level voltage in volts

The above keyword references are designed to aid 2920 program debugging.

ISIS Compatibilities

The 2920 software simulator runs under the ISIS "submit" facility. The 2920 software simulator uses the ISIS-II line editing capabilities to correct errors in an input line on the Intellec console.

Sample 2920 Simulation Session

```
-SM2920
*LIST SAW.LOG     ; SAVE OUTPUT FROM THIS SESSION IN LOG FILE
*LOAD SAW.HEX     ; LOAD SAWTOOTH GENERATOR ASSEMBLED WITH A62920
*ROM 0 TO 5     ; EXAMINE FIRST PART OF PROGRAM
ROM 000 = LDA .Y,.Y,R00,IN0
ROM 001 = LDA .Y,.Y,R00,IN0
ROM 002 = LDA .Y,.Y,R00,IN0
ROM 003 = SUB .Y,KP1,R00,IN0
ROM 004 = SUB .Y,KP1,R01,IN0
ROM 005 = LDA DAR,.Y,R00,IN0
*IN0 = SIN(4000*TIME)     ; SPECIFY INPUT SIGNAL TO BE SIMULATED
*QUALIFIER = PC=0     ; COLLECT TRACE ONLY ONCE PER PROGRAM PASS
*TRACE = TIME, IN0, OUT0     ; TRACE THESE THREE ITEMS
*SIMULATE FROM 0 TILL TIME>TPI/4000     ; SIMULATE FOR A WHILE
   TIME                IN0                 OUT0
SIMULATION BEGUN
     0 00009600          0.37463213          0.25000000
     0 00019200          0.69469800          0.06250000
     0 00028800          0.91357910          0.75000000
     0 00038400          0.99939463          0.56250000
     0 00048000          0.93964551          0.37500000
     0 00057600          0.74303413          0.18750000
     0 00067200          0.43819771          0.00000000
     0 00076800          0.06953646          0.68750000
     0 00086400         -0.30925317          0.00000000
     0 00096000         -0.64299883          0.00000000
     0 00105600         -0.88308994          0.00000000
     0 00115200         -0.99455645          0.00000000
     0 00124800         -0.96116289          0.00000000
     0 00134400         -0.78777319          0.00000000
     0 00144000         -0.49964165          0.00000000
     0 00153600         -0.13873627          0.00000000
SIMULATION TERMINATED
*: PROGRAM WORKS
*EXIT
```

SPECIFICATIONS

Operating Equipment

Required Hardware

Intellec® Microcomputer Development System
—Model 800 or 888
—Series II Model 220, 225, 230
64K Bytes of RAM Memory
One or two Floppy Disk Drives
—Single or Double Density
System Console
—CRT or interactive hard copy device

Required Software

ISIS-II Diskette Operating System

Optional Hardware

Line Printer
iSBC-310 High-Speed Mathematics Unit
Universal PROM Programmer

Optional Software

FORTRAN-80 (Product Code MDS-301)

Documentation Package

2920 Assembly User's Guide (9800987)
2920 Simulator User's Guide (9800988)
2920 Signal Processing Application Compiler User's Guide (121529)

Shipping Media

Flexible Diskettes
—Single and Double Density

ORDERING INFORMATION

Product Code	Description
MCI-20-SPS	2920 Software Support Package Includes 2920 Signal Processing Application Compiler and 2920 Assembler/Simulator Software