Casting Emission Reduction Program

Prepared by:

TECHNIKON LLC

5301 Price Avenue ▼ McClellan, CA, 95652 ▼ (916) 929-8001

www.technikonllc.com

US Army Task N256

Phase 1 SIVL Executive Software and Report

Technikon #1256-GSA R3.3

This document was revised for unlimited public distribution and published on

6 May 2003

EPA United States Environmental Protection Agency

CISA

AFS AMERICAN FOUNDRY SOCIETY, INC.

USCAR UNITED STATES COUNCIL FOR AUTOMOTIVE RESEARCH

DaimlerChrysler Ford Motor Company General Motors

SYSTEM INTEGRATION AND VALIDATION LABORATORY (SIVL)

DEVELOPMENT STATUS

ENGINEERING REPORT

GSA R3.3

Phase 1 SIVL Executive Software and Report, Communicating with SCPI Instrument

Technikon LLC

SYSTEM INTEGRATION AND VALIDATION LABORATORY (SIVL) DEVELOPMENT STATUS ENGINEERING REPORT

CONTRACT NO. GS-10P-99-EBD-0005

SUB 000800313

Task ABH 210-191

9 October 2000

Prepared for:

Concurrent Technologies Corporation
475 14th St., Suite 550
Oakland, CA 94612
Attn: Mike Walton, Program Manager

Prepared by:
Technikon LLC
5301 Price Ave.
McClellan, CA 95652

INTRODUCTION

This report provides a technical summary of the System Integration and Validation Laboratory Phase I development status.

Task Objective

Phase 1 System Integration and Validation Laboratory (SIVL) Test Executive software and report documenting communication with simple SCPI instrument.

System Integration and Validation Laboratory (SIVL)

Technikon performed work on the SIVL which included developing instrument validation software that could be used to validate a simple SCPI instrument. The instrument which was chosen for this task is a an HP34401A SCPI Digital Multimeter (DMM). In order to use this software, an instrument model of the DMM was developed to use with the CERP instrument simulator. The attached files include:

File Name	Contents	Notes
ui.zip	validation test program	includes data files with different validation tests
multimeter.zip	multimeter test application	Used for testing the validation softwar or other software
dmm-model 1.2.zip	dmm model	data files for DMM instrument model, works with instrument simulator versi 1.2 or 1.3

The validation test program is the SIVL test executive which is used to run a validation test of a real or simulated instrument. It has the capability of reading a test data file containing the test, and then running the test on an instrument. The test file contains SCPI commands and expected responses for an instrument. If a response is not within a defined tolerance of the expected result, an error is written to the log file. A summary of the results including the total errors is presented at the end of the test.

The multimeter program is a simple test program that can be used to test different client applications. It allows the client program to connect to it and send it ASCII SCPI commands, and prints out each command it receives. The user can type in a response to the command if desired, and the response will be send back to the client program.

The dmm model contains two data files which are used with the SIVL instrument simulator to perform a simulation of a SCPI emission bench instrument. One of the files defines the SCPI commands which are accepted by the model, and the other file defines the model behavior. A copy of the DMM model is included in Appendix I.

Appendix I

DMM Model

Version 1.2

DMM SDF data file. This file defines all of the SCPI commands which are supported by the DMM model, and maps them to simulator routines (defined in the DMM file).

```
:ABOR->NoResponse
ARM:SOUR->TriggerSource
ARM:COUN->TriggerCount
ARM:SEQ:SOUR->TriggerSource
ARM:SEQ:COUN->TriggerCount
:ARM:SEQ:SOUR->TriggerSource
:ARM:SEQ:COUN->TriggerCount
ARM:LAY:SOUR->TriggerSource
ARM:LAY:COUN->TriggerCount
ARM:SEQ:LAY:SOUR->TriggerSource
ARM:SEQ:LAY:COUN->TriggerCount
:CONF?->GetConfiguration
:CONF->SetVoltDC
CONF:VOLT->SetVoltDC
CONF:VOLT:DC->SetVoltDC
CONF:VOLT:AC->SetVoltAC
CONF:CURR->SetCurrentDC
CONF:CURR:DC->SetCurrentDC
CONF:CURR:AC->SetCurrentAC
CONF:RES->SetResistance
CONF:FRES->SetFResistance
:CONF:VOLT:DC->SetVoltDC
:CONF:VOLT:AC->SetVoltAC
:CONF:CURR->SetCurrentDC
:CONF:CURR:DC->SetCurrentDC
:CONF:CURR:AC->SetCurrentAC
:CONF:RES->SetResistance
:CONF:FRES->SetFResistance
CONF:SCAL:VOLT->SetVoltDC
CONF:SCAL:VOLT:DC->SetVoltDC
CONF:SCAL:VOLT:AC->SetVoltAC
CONF:SCAL:CURR->SetCurrentDC
CONF:SCAL:CURR:DC->SetCurrentDC
CONF:SCAL:CURR:AC->SetCurrentAC
CONF:SCAL:RES->SetResistance
CONF:SCAL:FRES->SetFResistance
:FETC?->ReturnVoltDC
FETC:VOLT?->ReturnVoltDC
FETC:VOLT:DC?->ReturnVoltDC
FETC:VOLT:AC?->ReturnVoltAC
FETC:CURR?->ReturnCurrDC
FETC:CURR:DC?->ReturnCurrDC
FETC:CURR:AC?->ReturnCurrAC
FETC:RES?->ReturnRes
FETC:FRES?->ReturnFRes
```

```
:FETC:VOLT:DC?->ReturnVoltDC
:FETC:VOLT:AC?->ReturnVoltAC
:FETC:CURR?->ReturnCurrDC
:FETC:CURR:DC?->ReturnCurrDC
:FETC:CURR:AC?->ReturnCurrAC
:FETC:RES?->ReturnRes
:FETC:FRES?->ReturnFRes
FETC:SCAL:VOLT?->ReturnVoltDC
FETC:SCAL:VOLT:DC?->ReturnVoltDC
FETC:SCAL:VOLT:AC?->ReturnVoltAC
FETC:SCAL:CURR?->ReturnCurrDC
FETC:SCAL:CURR:DC?->ReturnCurrDC
FETC:SCAL:CURR:AC?->ReturnCurrAC
FETC:SCAL:RES?->ReturnRes
FETC:SCAL:FRES?->ReturnFRes
:INIT->Initiate
INIT:IMM->Initiate
INIT:IMM:ALL->Initiate
:MEAS?->ReturnVoltDC
MEAS:VOLT?->ReturnVoltDC
MEAS:VOLT:DC?->ReturnVoltDC
MEAS:VOLT:AC?->ReturnVoltAC
MEAS:CURR?->ReturnCurrDC
MEAS:CURR:DC?->ReturnCurrDC
MEAS:CURR:AC?->ReturnCurrAC
MEAS:RES?->ReturnRes
MEAS:FRES?->ReturnFRes
:MEAS:VOLT:DC?->ReturnVoltDC
:MEAS:VOLT:AC?->ReturnVoltAC
:MEAS:CURR?->ReturnCurrDC
:MEAS:CURR:DC?->ReturnCurrDC
:MEAS:CURR:AC?->ReturnCurrAC
:MEAS:RES?->ReturnRes
:MEAS:FRES?->ReturnFRes
MEAS:SCAL:Voltage?->ReturnVoltDC
MEAS:SCAL:VOLT:DC?->ReturnVoltDC
MEAS:SCAL:VOLT:AC?->ReturnVoltAC
MEAS:SCAL:CURR?->ReturnCurrDC
MEAS:SCAL:CURR:DC?->ReturnCurrDC
MEAS:SCAL:CURR:AC?->ReturnCurrAC
MEAS:SCAL:RES?->ReturnRes
MEAS:SCAL:FRES?->ReturnFRes
:READ?->ReturnVoltDC
READ:VOLT?->ReturnVoltDC
READ:VOLT:DC?->ReturnVoltDC
READ:VOLT:AC?->ReturnVoltAC
READ:CURR?->ReturnCurrDC
```

```
READ:CURR:DC?->ReturnCurrDC
READ:CURR:AC?->ReturnCurrAC
READ:RES?->ReturnRes
READ:FRES?->ReturnFRes
:READ:VOLT:DC?->ReturnVoltDC
:READ:VOLT:AC?->ReturnVoltAC
:READ:CURR?->ReturnCurrDC
:READ:CURR:DC?->ReturnCurrDC
:READ:CURR:AC?->ReturnCurrAC
:READ:RES?->ReturnRes
:READ:FRES?->ReturnFRes
READ:SCAL:VOLT?->ReturnVoltDC
READ:SCAL:VOLT:DC?->ReturnVoltDC
READ:SCAL:VOLT:AC?->ReturnVoltAC
READ:SCAL:CURR?->ReturnCurrDC
READ:SCAL:CURR:DC?->ReturnCurrDC
READ:SCAL:CURR:AC?->ReturnCurrAC
READ:SCAL:RES?->ReturnRes
READ:SCAL:FRES?->ReturnFRes
ROUT:TERM?->ReturnFront
:ROUT:TERM?->ReturnFront
SENS:FUNC->SetFunction
SENS:FUNC?->GetFunction
:SENS:FUNC?->GetFunction
SENS:FUNC:ON->SetFunction
SENS:FUNC:ON?->GetConfiguration
SENS:VOLT:RANG->SetVDCRange
SENS:VOLT:RANG?->GetVDCRange
SENS:VOLT:DC:RANG->SetVDCRange
SENS:VOLT:DC:RANG?->GetVDCRange
SENS:VOLT:AC:RANG->SetVACRange
SENS:VOLT:AC:RANG?->GetVACRange
SENS:CURR:RANG->SetCDCRange
SENS:CURR:RANG?->GetCDCRange
SENS:CURR:DC:RANG->SetCDCRange
SENS:CURR:DC:RANG?->GetCDCRange
SENS:CURR:AC:RANG->SetCACRange
SENS:CURR:AC:RANG?->GetCACRange
SENS:RES:RANG->SetRESRange
SENS:RES:RANG?->GetRESRange
SENS:FRES:RANG->SetFRESRange
SENS:FRES:RANG?->GetFRESRange
:SENS:VOLT:DC:RANG?->SetVDCRange
:SENS:VOLT:AC:RANG?->GetVACRange
:SENS:CURR:RANG?->GetCDCRange
:SENS:CURR:DC:RANG?->GetCDCRange
:SENS:CURR:AC:RANG?->GetCACRange
```

```
:SENS:RES:RANG?->GetRESRange
:SENS:FRES:RANG?->GetFRESRange
SENS:VOLT:RANG:AUTO->SetRangeAuto
SENS:VOLT:DC:RANG:AUTO->SetRangeAuto
SENS:VOLT:DC:RANG:AUTO?->GetRangeAuto
SENS:VOLT:AC:RANG:AUTO->SetRangeAuto
SENS:CURR:RANG:AUTO->SetRangeAuto
SENS:CURR:DC:RANG:AUTO->SetRangeAuto
SENS:CURR:AC:RANG:AUTO->SetRangeAuto
SENS:RES:RANG:AUTO->SetRangeAuto
SENS:FRES:RANG:AUTO->SetRangeAuto
:SENS:VOLT:DC:RANG:AUTO?->GetRangeAuto
:SENS:VOLT:AC:RANG:AUTO?->GetRangeAuto
:SENS:CURR:RANG:AUTO?->GetRangeAuto
:SENS:CURR:DC:RANG:AUTO?->GetRangeAuto
:SENS:CURR:AC:RANG:AUTO?->GetRangeAuto
:SENS:RES:RANG:AUTO?->GetRangeAuto
:SENS:FRES:RANG:AUTO?->GetRangeAuto
SENS:VOLT:RES->SetVDCRes
SENS:VOLT:RES?->GetVDCRes
SENS:VOLT:DC:RES->SetVDCRes
SENS:VOLT:DC:RES?->GetVDCRes
SENS:VOLT:AC:RES->SetVACRes
SENS:VOLT:AC:RES?->GetVACRes
SENS:CURR:RES->SetCDCRes
SENS:CURR:RES?->GetCDCRes
SENS:CURR:DC:RES->SetCDCRes
SENS:CURR:DC:RES?->GetCDCRes
SENS:CURR:AC:RES->SetCACRes
SENS:CURR:AC:RES?->GetCACRes
SENS:RES:RES->SetRESRes
SENS:RES:RES?->GetRESRes
SENS:FRES:RES->SetFRESRes
SENS:FRES:RES?->GetFRESRes
:SENS:VOLT:RES?->GetVDCRes
:SENS:VOLT:DC:RES?->GetVDCRes
:SENS:VOLT:AC:RES?->GetVACRes
:SENS:CURR:RES?->GetCDCRes
:SENS:CURR:DC:RES?->GetCDCRes
:SENS:CURR:AC:RES?->GetCACRes
:SENS:RES:RES?->GetRESRes
:SENS:FRES:RES?->GetFRESRes
SENS:RES:OCOM->SetOCOM
SENS:RES:OCOM?->GetOCOM
:SENS:RES:OCOM?->GetOCOM
SENS:FRES:OCOM->SetOCOM
SENS:FRES:OCOM?->GetOCOM
```

```
:SENS:FRES:OCOM?->GetOCOM
:TRIG->NoResponse
TRIG:COUN->TriggerCount
:TRIG:COUN->TriggerCount
TRIG:DEL->TriggerDelay
TRIG:SOUR->TriggerSource
TRIG:SEQ:COUN->TriggerCount
:TRIG:SEQ:COUN->TriggerCount
TRIG:SEQ1:COUN->TriggerCount
TRIG:SEQ2:COUN->TriggerCount
TRIG:SEQ:DEL->TriggerDelay
:TRIG:SEQ:DEL->TriggerDelay
TRIG:SEQ1:DEL->TriggerDelay
TRIG:SEQ2:DEL->TriggerDelay
TRIG:SEQ:SOUR->TriggerSource
:TRIG:SEQ:SOUR->TriggerSource
TRIG:SEQ1:SOUR->TriggerSource
TRIG:SEQ2:SOUR->TriggerSource
SYST:ERR?->ReturnNoError
:SYST:ERR?->ReturnNoError
SYST:ERR:ALL?->ReturnNoError
SYST:ERR:CODE?->ReturnNoError
SYST:ERR:CODE:ALL?->ReturnNoError
SYST:ERR:CODE:NEXT?->ReturnNoError
SYST:ERR:NEXT?->ReturnNoError
SYST:ERR:ENABle->SetErrList
SYST:ERR:ENABle:ADD->SetErrList
SYST:ERR:ENABle:DEL->SetErrList
SYST:ERR:ENABle:LIST->SetErrList
:SYST:ERR:ENABle:LIST->SetErrList
SYST:LOCK:REL->NoResponse
SYST:LOCK:REQ?->ReturnOne
SYST:LOCK:OWN?->ReturnOwner
:SYST:LOCK:REL->NoResponse
:SYST:LOCK:REQ?->ReturnOne
:SYST:LOCK:OWN?->ReturnOwner
SYST:REM->NoResponse
:SYST:REM->NoResponse
SYST:DATE->SetDate
SYST:DATE?->BenchDate
:SYST:DATE->SetDate
:SYST:DATE?->BenchDate
:SYST:TIME->SetTime
SYST:TIME?->BenchTime
SYST:TIME:TIMer->SetTimerState
:SYST:TIME?->BenchTime
:SYST:TIME:TIMer->SetTimerState
```

```
:SYST:TIME:TIMer:COUN->SetTimerCount
SYST:TIME:TIMer:STAT->SetTimerState
STAT:OPER?->StatusOperation
:STAT:OPER?->StatusOperation
STAT:OPER:COND?->StatusOperation
:STAT:OPER:COND?->StatusOperation
STAT:OPER:ENAB->SetEnableList
:STAT:OPER:ENAB->SetEnableList
STAT:OPER:ENAB?->GetEnableList
STAT:OPER:EVEN?->StatusOperation
:STAT:OPER:EVEN?->StatusOperation
STAT:QUES?->StatusOperation
:STAT:QUES?->StatusOperation
STAT:QUES:COND?->StatusOperation
:STAT:QUES:COND?->StatusOperation
STAT:QUES:ENAB->SetEnableList
:STAT:QUES:ENAB->SetEnableList
STAT:QUES:ENAB?->GetEnableList
STAT:QUES:EVEN?->StatusOperation
:STAT:QUES:EVEN?->StatusOperation
*RST->Reset
*TST?->ReturnPlusZero
*WAI->WaitForComplete
```

DMM DDL file. This file defines the functionality supported by the DMM model. It uses the Instrument Definition Language (InDL) defined in the Instrument Simulator InDL reference document.

```
Instrument BagBench

Var
     CONC1 : NUMERIC;
     CONC2 : NUMERIC;
     CONC3 : NUMERIC;
     CONC4 : NUMERIC;
     SDEV1 : NUMERIC;
     SDEV2 : NUMERIC;
     SDEV3 : NUMERIC;
     SDEV4 : NUMERIC;
     DATA1 : NUMERIC;
     DATA2 : NUMERIC;
     DATA3 : NUMERIC;
     DATA4 : NUMERIC;
     TAB1 : NUMERIC;
     TAB2 : NUMERIC;
     TAB3 : NUMERIC;
     TAB4 : NUMERIC;
     TEMPF : NUMERIC;
     CONFIG : INTEGER;
     temp : NUMERIC;
     VDCRange: NUMERIC;
     VDCRes: NUMERIC;
     VACRange: NUMERIC;
     VACRes: NUMERIC;
     CDCRange: NUMERIC;
     CACRange: NUMERIC;
     CDCRes: NUMERIC;
     CACRes: NUMERIC;
     RESRange: NUMERIC;
     RESRes: NUMERIC;
     FRESRange: NUMERIC;
     FRESRes: NUMERIC;
     STATUS : NUMERIC;
     Seed : INTEGER;
     WORKING : BOOLEAN;
     RANGEAUTO : BOOLEAN;
     OCOM : BOOLEAN;
     TimerCount : INTEGER;
     TriggerCount : INTEGER;
     ReturnCount : INTEGER;
     ErrEnableList : STRING;
     StatEnableList : STRING;
```

```
     DateString : STRING;
     TimeString : STRING;
     TimerState : INTEGER;
     OK : INTEGER;
End Var

     Event BenchDate
       RESPOND DATE;
     End Event

     Event SetDate
     (
          term1 : STRING
     )
          DateString := term1;
     End Event

     Event SetTime
     (
          term1 : STRING
     )
          TimeString := term1;
     End Event

     Event SetTimerCount
     (
          term1 : INTEGER
     )
          TimerCount := term1;
     End Event

     Event BenchTime
       RESPOND TIME;
     End Event

     Event SetTimerState
     (
          term1 : STRING
     )
          IF term1 == "ON" THEN
               TimerState := 1;
          ELSE
               TimerState := 0;
          END IF
     End Event
```

```
Event NoResponse
     WORKING :=TRUE;
End Event

Event UnknownResponse
     RESPOND 0;
End Event

Event ReturnZero
     RESPOND 0;
End Event

Event ReturnOne
     RESPOND 1;
End Event

Event SetFunction
(
     term1 : STRING
)
     IF term1=="VOLT:AC" THEN
          CONFIG:=2;
     END IF
     IF term1=="CURR:DC" THEN
          CONFIG:=3;
     END IF
     IF term1=="CURR:AC" THEN
          CONFIG:=4;
     END IF
     IF term1=="RES" THEN
          CONFIG:=5;
     END IF
     IF term1=="FRES" THEN
          CONFIG:=6;
     END IF
     IF term1=="VOLT:DC" THEN
          CONFIG:=1;
     END IF
End Event

Event GetVDCRange
     RESPOND VDCRange;
End Event

Event GetVDCRes
     RESPOND VDCRes;
End Event
```

```
Event GetVACRange
     RESPOND VACRange;
End Event

Event GetVACRes
     RESPOND VACRes;
End Event

Event GetCDCRange
     RESPOND CDCRange;
End Event

Event GetCDCRes
     RESPOND CDCRes;
End Event

Event GetFRESRange
     RESPOND FRESRange;
End Event

Event GetFRESRes
     RESPOND FRESRes;
End Event

Event GetCACRange
     RESPOND CACRange;
End Event

Event GetCACRes
     RESPOND CACRes;
End Event

Event GetRESRange
     RESPOND RESRange;
End Event

Event GetRESRes
     RESPOND RESRes;
End Event

Event SetRangeAuto
(
     term1 : STRING
)
     IF term1 == "ON" THEN
          RANGEAUTO := TRUE;
```

```
     ELSE
          RANGEAUTO := FALSE;
     END IF
End Event

Event GetRangeAuto
     IF RANGEAUTO == TRUE THEN
          RESPOND 1;
     ELSE
          RESPOND 0;
     END IF
End Event

Event GetFunction

     IF CONFIG == 0 THEN
          RESPOND "VOLTDC";
     END IF
     IF CONFIG == 1 THEN
          RESPOND "VOLTDC";
     END IF
     IF CONFIG == 2 THEN
          RESPOND "VOLTAC";
     END IF
     IF CONFIG == 3 THEN
          RESPOND "CurrentDC";
     END IF
     IF CONFIG == 4 THEN
          RESPOND "CurrentAC";
     END IF
     IF CONFIG == 5 THEN
          RESPOND "Resistance";
     END IF
     IF CONFIG == 6 THEN
          RESPOND "FResistance";
     END IF

End Event

Event GetConfiguration

     IF CONFIG == 0 THEN
          RESPOND "VOLT",VDCRange,VDCRes;
     END IF
     IF CONFIG == 1 THEN
          RESPOND "VOLTDC",VDCRange,VDCRes;
     END IF
```

```
     IF CONFIG == 2 THEN
          RESPOND "VOLTAC",VACRange,VACRes;
     END IF
     IF CONFIG == 3 THEN
          RESPOND "CurrentDC",CDCRange,CDCRes;
     END IF
     IF CONFIG == 4 THEN
          RESPOND "CurrentAC",CACRange,CACRes;
     END IF
     IF CONFIG == 5 THEN
          RESPOND "Resistance",RESRange,RESRes;
     END IF
     IF CONFIG == 6 THEN
          RESPOND "FResistance",FRESRange,FRESRes;
     END IF

End Event

Event GetOCOM
     IF OCOM == TRUE THEN
          RESPOND 1;
     ELSE
          RESPOND 0;
     END IF
End Event

Event SetOCOM
(
     term1 : STRING
)
     IF term1 == "ON" THEN
          OCOM := TRUE;
     ELSE
          OCOM := FALSE;
     END IF
End Event

Event Reset
     CONFIG := 0;
End Event

Event SetErrList
(
     term1 : STRING
)
     ErrEnableList := term1;
End Event
```

```
Event SetEnableList
(
     term1 : STRING
)
     StatEnableList := term1;
End Event

Event GetEnableList
     RESPOND StatEnableList;
End Event


Event SetVoltDC
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )
     CONFIG := 1;
     VDCRange := term1;
     VDCRes := term2;
End Event

Event SetVoltAC
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )
     CONFIG := 2;
     VACRange := term1;
     VACRes := term2;
End Event

Event SetCurrentDC
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )
     CONFIG := 3;
     CDCRange := term1;
     CDCRes := term2;
End Event

Event SetCurrentAC
  (
     term1 : NUMERIC,
     term2 : NUMERIC
```

```
  )
     CONFIG := 4;
     CACRange := term1;
     CACRes := term2;
End Event

Event SetResistance
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )
     CONFIG := 5;
     RESRange := term1;
     RESRes := term2;
End Event

Event SetFResistance
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )
     CONFIG := 6;
     FRESRange := term1;
     FRESRes := term2;
End Event

Event SetVDCRange
  (
     term1 : NUMERIC
  )
     CONFIG := 1;
     VDCRange := term1;
End Event

Event SetVACRange
  (
     term1 : NUMERIC
  )
     CONFIG := 2;
     VACRange := term1;
End Event

Event SetCDCRange
  (
     term1 : NUMERIC
  )
     CONFIG := 3;
```

```
     CDCRange := term1;
End Event

Event SetCACRange
  (
     term1 : NUMERIC
  )
     CONFIG := 4;
     CACRange := term1;
End Event

Event SetRESRange
  (
     term1 : NUMERIC
  )
     CONFIG := 5;
     RESRange := term1;
End Event

Event SetFRESRange
  (
     term1 : NUMERIC
  )
     CONFIG := 6;
     FRESRange := term1;
End Event

Event SetVDCRes
  (
     term1 : NUMERIC
  )
     CONFIG := 1;
     VDCRes := term1;
End Event

Event SetVACRes
  (
     term1 : NUMERIC
  )
     CONFIG := 2;
     VACRes := term1;
End Event

Event SetCDCRes
  (
     term1 : NUMERIC
  )
```

```
     CONFIG := 3;
     CDCRes := term1;
End Event

Event SetCACRes
  (
     term1 : NUMERIC
  )
     CONFIG := 4;
     CACRes := term1;
End Event

Event SetRESRes
  (
     term1 : NUMERIC
  )
     CONFIG := 5;
     RESRes := term1;
End Event

Event SetFRESRes
  (
     term1 : NUMERIC
  )
     CONFIG := 6;
     FRESRes := term1;
End Event

Event ReturnFloat

     Seed := RANDOM;

     TEMPF := SIN(Seed);
     RESPOND TEMPF;
End Event


Event ReturnVoltDC
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )

     Seed := RANDOM;
     IF VDCRange == 0 THEN
       TEMPF := SIN(Seed) * 5.0;
     ELSE
```

```
        TEMPF := SIN(Seed) * VDCRange;
      END IF
      IF ReturnCount == 0 THEN
           RESPOND TEMPF;
      END IF
      IF ReturnCount == 1 THEN
           RESPOND TEMPF;
      END IF
      IF ReturnCount == 2 THEN
           RESPOND TEMPF,TEMPF;
      END IF
      IF ReturnCount == 3 THEN
           RESPOND TEMPF,TEMPF,TEMPF;
      END IF
      IF ReturnCount == 4 THEN
           RESPOND TEMPF,TEMPF,TEMPF,TEMPF;
      END IF
      IF ReturnCount == 5 THEN
           RESPOND TEMPF,TEMPF,TEMPF,TEMPF,TEMPF;
      END IF
      ReturnCount := 0;
End Event

Event ReturnVoltAC
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )
     Seed := RANDOM;
     IF VACRange == 0 then
       TEMPF := SIN(Seed) * 10.0;
     ELSE
       TEMPF := SIN(Seed) * VACRange;
     END IF

     IF ReturnCount == 0 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 1 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 2 THEN
          RESPOND TEMPF,TEMPF;
     END IF
     IF ReturnCount == 3 THEN
          RESPOND TEMPF,TEMPF,TEMPF;
     END IF
```

```
     IF ReturnCount == 4 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 5 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     ReturnCount := 0;
End Event

Event ReturnCurrDC
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )

     Seed := RANDOM;
     IF CDCRange == 0 then
       TEMPF := SIN(Seed) * 15.0;
     ELSE
       TEMPF := SIN(Seed) * CDCRange;
     END IF

     IF ReturnCount == 0 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 1 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 2 THEN
          RESPOND TEMPF,TEMPF;
     END IF
     IF ReturnCount == 3 THEN
          RESPOND TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 4 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 5 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     ReturnCount := 0;
End Event

Event ReturnCurrAC
  (
     term1 : NUMERIC,
     term2 : NUMERIC
```

```
  )

     Seed := RANDOM;
     IF CACRange == 0 then
       TEMPF := SIN(Seed) * 20.0;
     ELSE
       TEMPF := SIN(Seed) * CACRange;
     END IF

     IF ReturnCount == 0 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 1 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 2 THEN
          RESPOND TEMPF,TEMPF;
     END IF
     IF ReturnCount == 3 THEN
          RESPOND TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 4 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 5 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     ReturnCount := 0;
End Event

Event ReturnRes
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )
     Seed := RANDOM;

     Seed := RANDOM;
     IF RESRange == 0 then
       TEMPF := SIN(Seed) * 100.0;
     ELSE
       TEMPF := SIN(Seed) * RESRange;
     END IF

     IF ReturnCount == 0 THEN
          RESPOND TEMPF;
     END IF
```

```
     IF ReturnCount == 1 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 2 THEN
          RESPOND TEMPF,TEMPF;
     END IF
     IF ReturnCount == 3 THEN
          RESPOND TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 4 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 5 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     ReturnCount := 0;
End Event

Event ReturnFRes
  (
     term1 : NUMERIC,
     term2 : NUMERIC
  )

     Seed := RANDOM;
     IF FRESRange == 0 then
       TEMPF := SIN(Seed) * 100.0;
     ELSE
       TEMPF := SIN(Seed) * FRESRange;
     END IF

     IF ReturnCount == 0 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 1 THEN
          RESPOND TEMPF;
     END IF
     IF ReturnCount == 2 THEN
          RESPOND TEMPF,TEMPF;
     END IF
     IF ReturnCount == 3 THEN
          RESPOND TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 4 THEN
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     IF ReturnCount == 5 THEN
```

```
          RESPOND TEMPF,TEMPF,TEMPF,TEMPF,TEMPF;
     END IF
     ReturnCount := 0;
End Event

Event StatusOperation

     Seed := RANDOM;
     IF Seed < 25000 THEN
          RESPOND 0;
     ELSE
          RESPOND "10101";
     END IF
End Event


Event SetOneorZero

     Seed := RANDOM;
     IF Seed < 25000 THEN
          RESPOND "1";
     ELSE
          RESPOND 0;
     END IF
End Event

Event GetInteger
     RESPOND "32767";
End Event

Event ReturnOwner
     RESPOND "AIGER";
End Event

Event ReturnNoError
     RESPOND "+0,No Error";
End Event

Event ReturnFront
     RESPOND "FRON";
End Event

Event ReturnPlusZero
     RESPOND "+0";
End Event

Event WaitForComplete
```

```
     WORKING :=TRUE;
End Event

Event TriggerSource
(
     term1 : STRING
  )
     WORKING :=TRUE;
End Event

Event TriggerCount
(
     term1 : INTEGER
  )
     TriggerCount := term1;
End Event

Event Initiate
     ReturnCount := TriggerCount;
End Event

Event TriggerDelay
(
     term1 : NUMERIC
 )
     WORKING :=TRUE;
End Event

Event SystCommSockFeedStart
     CONTINUOUS SystCommSockFeed;
End Event

Event SystCommSockFeedStop
     CONTINUOUS;
End Event

Event SystCommSockFeed
     Var
          Seed1 : NUMERIC;
          Seed2 : NUMERIC;
          PARM1 : NUMERIC;
          PARM2 : NUMERIC;
     End Var
Seed1 := RANDOM / RANDOM;
Seed2 := RANDOM / RANDOM;
PARM1 := Seed1 +19;
PARM2 := Seed2 +82;
```

```
     RESPOND PARM1 , PARM2;

     End Event

     Event SystCommSockFeedStart
          CONTINUOUS SystCommSockFeed;
     End Event

     Event SystCommSockFeedStop
          CONTINUOUS;
     End Event

     Event SystCommSockFeed
          Var
               Seed1 : NUMERIC;
               Seed2 : NUMERIC;
               PARM1 : NUMERIC;
               PARM2 : NUMERIC;
          End Var
     Seed1 := RANDOM / RANDOM;
     Seed2 := RANDOM / RANDOM;
     PARM1 := Seed1 +19;
     PARM2 := Seed2 +82;
     RESPOND PARM1 , PARM2;

     End Event

End Instrument
```