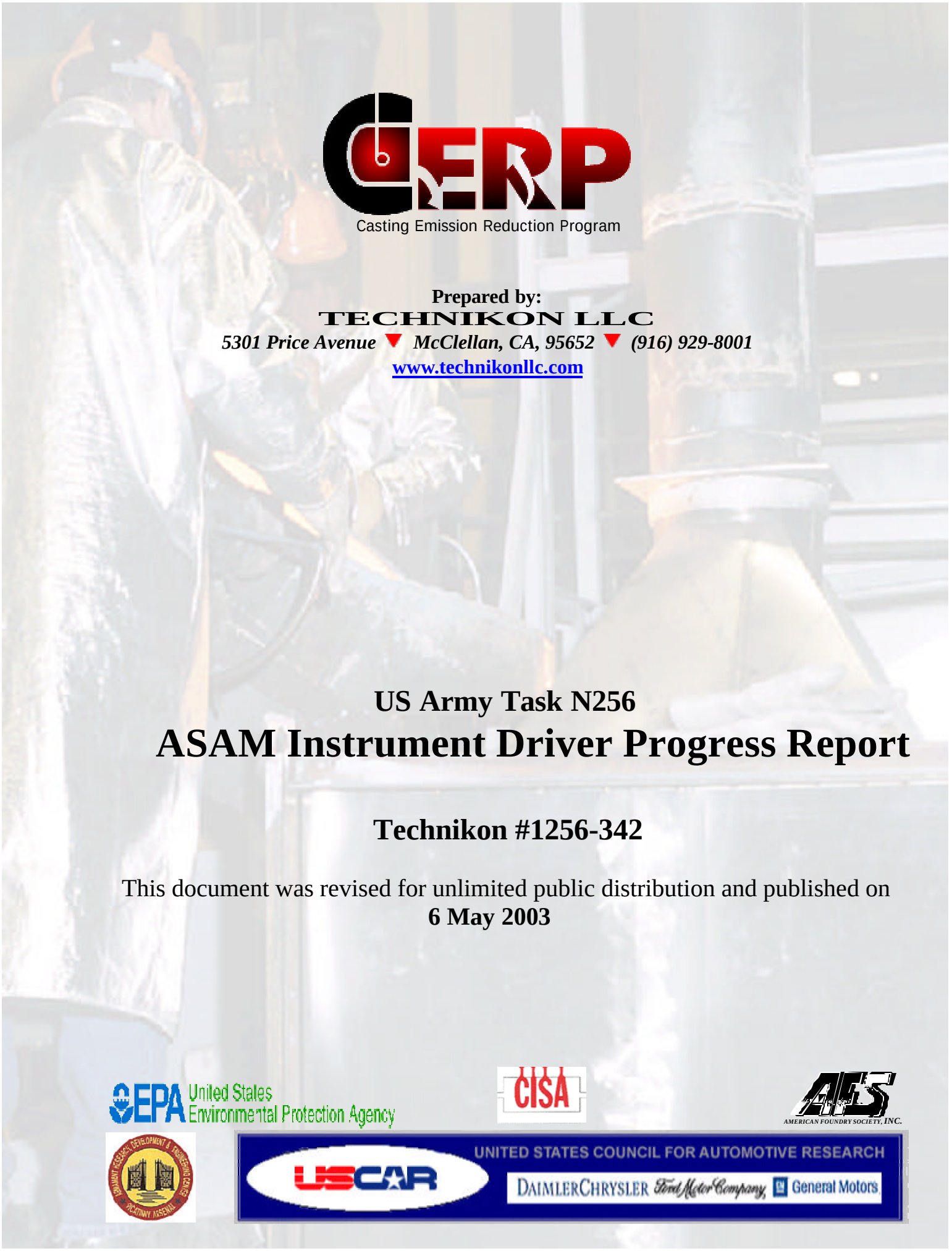

Casting Emission Reduction Program

Prepared by:

TECHNIKON LLC

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

www.technikonllc.com

US Army Task N256

ASAM Instrument Driver Progress Report

Technikon #1256-342

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

TECHNIKON
ENVIRONMENTAL DEVELOPMENT CENTER™

Association for the Standardization of Automation and Measuring Systems (ASAM) Instrument Driver Progress Report

WBS: 3.4.2

23 April 2001

TECHNIKON
ENVIRONMENTAL DEVELOPMENT CENTER™

ASAM Instrument Driver Progress Report
WBS 3.4.2

23 April 2001

Reviewed and Approved by: William C. Walden Date: 5 APR 01

A. Introduction

This report contains a summary of the progress to date on the Association for the Standarization of Automation and Measuring Systems (ASAM) Instrument Driver Development.

B. Background

The Emissions Bench Generic Device Interface (GDI) Driver was developed to provide a device-independent method of standard communication between ASAM compliant user applications and Standard Commands for Programmable Instruments (SCPI) compliant emission benches. The driver implements the ASAM API interface defined in the ASAM-GDI Part B, Specification of the Interface of an ASAM-GDI_Driver. The EBench API functions are shown in the include file GDI.H (See Appendix A). The standard SCPI-Ethernet interface to Emissions Bench devices is described in Standard Commands for Programmable Instruments, Volume 2 and Volume 4.

The following is a list of the API functions provided in the EBench GDI driver.

GDI_Attach
GDI_Initiate
GDI_Conclude
GDI_CreateFuncObject
GDI_DeleteFuncObject
GDI_CreateCommObject
GDI_DeleteCommObject
GDI_Execute
GDI_Read
GDI_Write
GDI_Cancel
GDI_Abort
GDI_Identify
GDI_Status
GDI_Information_Report
io_complete
io_event

These functions are standard for any ASAM GDI Driver and can be used by an application to perform driver-specific functions which are defined in a Device Capability Description (DCD) file. The format of a DCD is described in the ASAM-GDI Part A, Specification of the Device Capability Description ASAM-GDI_Driver.

The EBench specification consists of three Function Objects: 1) General EBench Sytem, 2) Analyzer Group (which consists of one or more analyzers), and 3) Analyzer. The EBench function objects provide the processing required for communication with actual Emissions Bench hardware. Function objects include parameters, attributes, and operations.

A list of the EBench-specific objects described in the EBench DCD (ebench_0_60.dcd) and implemented in the EBench GDI driver is included in Appendix B of this document.

In order to provide standard capabilities for any hardware device, the ASAM GDI standard defines the use of Virtual Devices (VDs). The Control VD is created to enable state transition and control of the device driver (Attach, Initiate, Status). The Device VD provides the function and function object capabilities (Create, Delete, Read, Write, and Execute). API calls populate the Device VD with one or more instances of all required device functions (Create Function Object). Required function objects are then attached to the function instance (Create Communication Object). Handles are assigned to each created virtual device, function, and function object in order to create a unique address for each instance of a function object. Once the VDs are created, initialized, and populated, and transition has been made to a legal state, API calls to Read/Write (for parameters or attributes) or Execute (for operations) can be made to send and receive commands and data from the EBench device.

C. Progress to Date

The Ebench Driver was developed to be compliant with the current version of the Ebench Device Capability Description (DCD), which is version 6.0. The EBench driver contains implementations of the following core capabilities:

- GDI API calls
- Control Virtual Devices
- State Transitions
- Device Virtual Devices
- Generic Function Control
- Generic Communication Object Control
- Generic Device Control

These capabilities are complete and tested. The results of these tests are included in the EBench GDI Driver Test Results Document.

In addition, the EBench driver contains implementation of the following EBench-specific capabilities:

- EBench Device communication and control
- Ebench System Function creation and control
  - Communication Object Create/Delete, Read/Write
  - Operation Execute
- Analyzer Group Function creation and control
  - Communication Object Create/Delete, Read/Write
  - Operation Execute
- Analyzer Function creation and control
  - Operation Execute

These capabilities are complete for Version 6.0 of the EBench specification defined in ebench_0_60.dcd and have been tested to the extent that test cases exist for these capabilities. The results of these tests are included in the EBench GDI Driver Test Results Document.

Capabilities which have not been fully implemented in the current version of the EBench GDI driver include:

- Asynchronous execution of all function objects
- Error checking, error handling, and error reporting

D. Summary

This report provided the current status of the ASAM Ebench Driver task. A prototype version of the driver has been implemented which is compliant with the current version of the Ebench specification. Additional work remains to be done as the specification is updated to the final version.

APPENDIX A

ASAM-GDI Device Driver Interfaces

GDI.H

```
/*****************************************************************************/
/*****************************************************************************/
/****                                                                     ****/
/****  Module: GDI.H for ASAM GDI driver                                  ****/
/****  Date:February 1st, 2001                                            ****/
/*****************************************************************************/
/*****************************************************************************/
/*****************************************************************************/

#include "Pa.h"

#ifndef GDI_H
/**   Defines Headerfile GDI.H @version 1.0   **/

#define GDI_H

//e.g  for creating a windows DLL
#define GDI_CALL          __declspec( dllexport )

#define GDI_CB __cdecl
//e.g  for creating call back style
//#define GDI_CB __cdecl

typedef short        (GDI_CB *GDI_CBP)();

#define NULL_GDI_CBP    (GDI_CBP)      0

#define NO_MOD_SEL      1

/*****************************************************************************/
/* Error                                                                     */
/*****************************************************************************/
/*****************************************************************************/
/* Invokation error                                                          */
/*****************************************************************************/

#define COM_ERR                 -1
#define NOTRUN                  -2
#define NOTINI          -3
#define DRVERR_OPF      -9
#define NOTASYNC        -12
#define NOTAVA                  -13
#define NOTIMP          -14
#define NOTEXE                  -15


/*****************************************************************************/
/* process error                                                             */
/*****************************************************************************/
#define  COM_ABO        0
#define  TRANS_ERR      1
#define  IOCPL_ERR      2
#define  KPAFU_ERR      3
#define IOHND_ERR       4
```

```
#define IOERR_OPEN      5
#define IOERR_WRITE     6
#define IOERR_READ      7
#define IOERR_EXECUTE           8

#define COM_OK                  0
#define COM_REJ                 1

/*****************************************************************************/
/* execution error                                                           */
/*****************************************************************************/
#define OTHER                   0
#define STATE_ERR               1

#define APP_UNR                 1

#define MODID_UNDEF 1
#define FUNCID_UNDEF            2
#define FUNMEM_UNDEF            3
#define DAT_ERROR       4
#define OBJ_EXIST               5

#define MEM_UNA                 1
#define PROC_UNA                2
#define CAP_UNA                 4
#define CONF_ERROR1     5
#define CONF_ERROR2     6

#define TIMEOUT                 1
#define DEADLOCK                2

#define NO_VDHND                1
#define NO_FUNCHND      2
#define NO_FUNCMEM      3
#define NO_OPID                 4
#define RO_OBJ          5
#define DOM_ERROR                6
#define TRANSIT_ERR              7
#define HW_ERR                          8

#define FCOMREQ                         1
#define JOBID_ERR                       1
#define CANC_NP                         2

/*****************************************************************************/
/* Call back errors                                                          */
/*****************************************************************************/
#define ACC_UNDEF               -1
#define ACC_TUNAV               -2

#define  COM_OK                                 0
#define  DEV_ERRORMSG                           1
#define  DEV_DATAERROR                          2
#define PROTOKOLLERROR                  3
#define  ERROR_INCOMMAND                1
#define  ERROR_INTELEGRAMM                      2
```

```
#define  ERROR_STATE                    3
#define  ERROR_DEVICEINREMOTE     4
#define  ERROR_CMDNOTALLOWED     5
#define  ERROR_NOTAVAILABLE              6
#define  ERROR_DATA                     7
#define  ERROR_FUNCNOTFOUND      6
#define  ERROR_UNKOWFUNCID               7

/*************************************************************************/
/* data types used by GDI API calls                                      */
/*************************************************************************/
typedef struct
{
        short  qual;
        short  grade;
        short  code;
        APIHND ptr;
} RESULT;

typedef struct
{
                long    DeviceVersion;
                char    *DriverName;
                long    DriverVersion;
                char    *Factory;
} IDENT;

typedef struct
{
                short int       log;
                short int       phys;
                short int       phase;
                RESULT                  detail;
} STAT;

/*************************************************************************/
/* types of call back functions supported by coordinator                 */
/*************************************************************************/
typedef short  (GDI_CB *GDI_CBP_CPL)(APIHND,RESULT*);
typedef short  (GDI_CB *GDI_CBP_IR)(APIHND,void*);
typedef short  (GDI_CB *GDI_CBP_EVENT)(APIHND,void*);

/*************************************************************************/
/* prototypes of GDI API function                                        */
/*************************************************************************/
#ifdef __cplusplus
extern "C" {
#endif  /* __cplusplus */

/* ------------------------------------------------------------------------- */
short int GDI_CALL GDI_Attach (GDI_CBP PtrCompl, GDI_CBP PtrAcc, GDI_CBP PtrInf);
/* ------------------------------------------------------------------------- */
/* ------------------------------------------------------------------------- */
short int GDI_CALL GDI_Initiate(APIHND ModId,
                        APIHND *VDHandle,
                        void    *IniDatPtr,
```

```
                        APIHND JobId,
                        RESULT *ResPtr);
/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */
short int GDI_CALL GDI_Conclude(APIHND VDHandle,APIHND JobId,RESULT *ResPtr);
/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */
short int GDI_CALL GDI_CreateFuncObject(APIHND VDHandle,
                            APIHND FuncName,
                            void   *CreatePar,
                            APIHND *FuncHandle,
                            APIHND JobId,
                            RESULT *ResPtr);

/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */
short int GDI_CALL GDI_DeleteFuncObject(APIHND VDHandle,
                            APIHND FuncHandle,
                            APIHND JobId, RESULT *ResPtr);
/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */
short int GDI_CALL GDI_CreateCommObject(APIHND VDHandle,
                            APIHND FuncHandle,
                            APIHND FuncMember,
                            APIHND GlobObjId,
                            APIHND JobId,
                            RESULT *ResPtr);
/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */
short int GDI_CALL GDI_DeleteCommObject(APIHND VDHandle,
                            APIHND FuncHandle,
                            APIHND FuncMember,
                            APIHND *GlobObjIdPtr,
                            APIHND JobId,
                            RESULT *ResPtr);
/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */
short int GDI_CALL GDI_Execute(APIHND VDHandle,
                        APIHND FuncHandle,
                        APIHND OpId,
                        void   *DatPtrW,
                        void   *DatPtrR,
                        APIHND JobId,
                        RESULT *ResPtr);
/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */
short int GDI_CALL GDI_Read(APIHND VDHandle,
                    APIHND FuncHandle,
                    APIHND FuncMember,
                    void   *DatPtr,
                    APIHND JobId,
                    RESULT *ResPtr);
/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */
short int GDI_CALL GDI_Write(APIHND VDHandle,
                     APIHND FuncHandle,
```

```
                    APIHND FuncMember,
                    void   *DatPtr,
                    APIHND JobId,
                    RESULT *ResPtr);
/* ------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------ */
short int GDI_CALL GDI_Cancel(APIHND VDHandle, APIHND JobId,          APIHND CanJob, RESULT *ResPtr);
/* ------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------ */
short int GDI_CALL GDI_Abort(APIHND VDHandle);
/* ------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------ */
short int GDI_CALL GDI_Identify(APIHND VDHandle,IDENT *idDatPtr,APIHND JobId, RESULT *ResPtr);
/* ------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------ */
short int GDI_CALL GDI_Status(APIHND VDHandle,STAT *StatPtr,APIHND  JobId, RESULT *ResPtr);
/* ------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------ */
short int PA_CB io_complete (APIHND  IO_Hnd, IO_STAT *io_status);
/* ------------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------------ */
short int PA_CB io_event (short intDevId,
// identifier of the periphery interface that has been passed by the platform adapter while opening.
        short int EventId,
// identifier of an event message
        void       *EventMsg);
//pointer to a single value or structure (type depends on the extension)
/* ------------------------------------------------------------------------------------ */
#ifdef __cplusplus
}
#endif
#endif
```

APPENDIX B

Functions and Function Objects

Defined in

EBench Specification Version 6.0

1. Function EbenchSystm

// Diagnostic objects
diagnosticHangup
diagnosticLeakCheck
diagnosticNoxEff
diagnosticCO2

// Trigger objects
InitiateImmediate
InitiateContinuous

// Communication objects
senseDataListType
DataChannelPorts
DataChannelAddress
socketConnect
socketClose
socketFeed

// General system objects
ebenchCleanDur
Clean
RouteOpen
systemDate
setSystemDate
systemError
systemLockReq
systemLockRel
systemLockOwner
systemTimer
wait
ErrorReport
statusRegisters
StatusRegisterMask

// Memory Tables
SetCPTable
GetCPTable
GasBottleTable
HUPResults
HUPTolerance
LeakCheckResults
LTolerance
GetLinSampleTable
SetAnalyzerLogTable
GetAnalyzerLogTable
NEResult
NETolerance
SetNOXEffLogTable
GetNOXEffLogTable
SdriftTable
ZdriftTable
SpanGas
TopGas

VerifyGas
ZeroGas

2. Function Analyzer_Group

// sense concentration setup objects
ConcCSet
ConcLower
ConcLSet
ConcTAlign
ConcUpper
sensCorrZDriftTol
sensCorrSDriftTol
senseCorrState
senseConcRange
StabilityParam
senseConcRangeAutoData

// Linearization objects
LinearizeAcquire
LinearizeAccept
LinearizeAuto
LinearizeCalculate
LinearizeCurve
linearizeVerify

// Sense objects
senseAverage
senseConcRangeAuto
senseStabilize
senseCorrectionAuto
senseCorrectionZeroAcquire
senseCorrectionSpanAcquire
senseCorrectionCalculate
senseFunction
senseData

3. Function Analyzer
getStatus
SetStatusMask