Goals for Device Interface Modules (DIMs)

A. Confining the impact of device changes on the software.

   1. User must not be able to do things that only work on one device.
   2. User must have no need to bypass the module.
   3. DIM should be minimal.

B. Simplifying the rest of the software.

C. Enforced disciplined use of resources (protocols).

D. Code sharing.

E. Efficient code (expert programmers).

The Abstract Interface Design Process

A. Iterate

   1. Find assumptions that are valid for all devices.
   2. Embody those assumptions in a set of access programs.
   3. Review for consistency.

B. Problem: Software designer not hardware expert.

   Solution: active design reviews

An Example: History of a simple module

A. A reasonable first draft

Figure 1: Excerpts from an Early Draft of the ADC Abstract Interface

Assumption List

1. The ADC provides a measure of barometric altitude, mach number, and true airspeed.
2. The above measurements are based on a common set of sensors. Therefore an inaccuracy in one ADC sensor may affect any of these outputs.
3. The ADC provides an indication if any of its sensors are not functioning properly.
4. The measurements are made assuming a sea level pressure of 29.92 inches of mercury.

Access Program Table

Program name	Parameter type	Parameter information
G_ADC_ALTITUDE	pl:distance;0	altitude assuming 29.92" sea level pressure
G_ADC_MACH_INDEX	pl:mach;0	mach
G_ADC_TRUE_AIRSPEED	pl:speed;0	true airspeed
G_ADC_FAIL_INDICATOR	pl:logical;0	**true** if ADC failed

An Example: History of a simple module

Problems with first draft:

1. Cannot use built-in test.
2. No report of failed-state values.
3. Ranges not specified.

An Example: History of a simple module

A later draft--sent for review

Assumption List

1. The ADC provides measurements of the barometric altitude, true airspeed and the mach number representation of the airspeed of the aircraft. Any known measurement errors are compensated for within the module. Altitude measurements are made assuming that the air pressure at sea level is 29.92" of mercury.
2. All of these measurements are based on a common set of sensors; therefore an inaccuracy in one ADC sensor will affect all measurements.
3. User programs are notified by means of an event when the ADC hardware fails. If the access programs for barometric altitude, true airspeed and mach number are called during an ADC failure, the last valid measurements (stale values) are provided.
4. The ADC is capable of performing a self-test upon command from the software. The result of this test is returned to the software.
5. The minimum measurable value for mach number and true airspeed is zero. The minimum barometric altitude measurable is fixed after system generation time, as are the maximum value and resolution for all measurements.

Access Program Table

Program name	Parameter type	Parameter information
G_ADC_BARO_ALTITUDE	pl:distance;0	corrected altitude assuming sea level pressure = 29.92" mercury
G_ADC_MACH_INDEX	pl:mach;0	corrected mach
G_ADC_RELIABILITY	pl:logical;0	true if ADC reliable
G_ADC_TRUE_AIRSPEED	pl:speed;0	corrected true airspeed
TEST_ADC	pl:logical;0	true if ADC passed self test

Event Table

Event	When signalled
@T (ADC unreliable)	When "ADC reliable" changes from true to false

An Example: History of a simple module

Problems with the later draft:

1. Sea level pressure correction must be internal.

2. Reliability of different outputs could be independent.

3. Minimum readings might not be 0.

An Example: History of a simple module

C. The published version:

Figure 3: Excerpts from the Final Version of the ADC Abstract Interface.

Assumption List

1. The ADC provides measurements of the barometric altitude, true airspeed, and the mach number representation of the airspeed of the aircraft (mach index). Any known measurement errors are compensated for within the module.

2. User programs are notified by means of events when one or more of the outputs are unavailable. A user program can also inquire about the reliability of individual outputs. f the access functions for barometric altitude, true airspeed, and mach number are called while the values are unreliable, the last valid measurements (stale values) are provided.

3. The ADC is capable of performing a self-test upon command from a user program. The result of this test is returned to the user program.

4. The minimum, maximum and resolution of all ADC measurements are fixed after system generation time.

5. The ADC will compute its outputs on the basis of a value for Sea Level Pressure (SLP) supplied to it by a user program. If no value is provided, SLP of 29.92 will be assumed.

Access Function Table

Program name	Parameter type	Parameter information
G_ADC_ALTITUDE	pl:distance;0	corrected altitude assuming SLP = 29.92 or user supplied SLP
	p2:logical;0	true if altitude valid
G_ADC_MACH_INDEX	p1:mach;0	corrected mach
	p2:logical;0	true if mach valid
G_ADC_TRUE_AIRSPEED	p1:speed;0	corrected true airspeed
	p2:logical;0	true if true airspeed valid
S_ADC_SLP	p1:pressure;I	sea level pressure
TEST_ADC	p1:logical;0	true if ADC passed self test

Event Table

Event	When signalled
@T(altitude invalid)	When “altitude valid” changes from true to false
@T(airspeed invalid)	When “true airspeed valid” changes from true to false
@T(mach invalid)	When “mach valid” changes from true to false

Design problems--Tradeoffs and compromises

A. Major variations among available devices

1. Sometimes differences are more than skin deep.
2. Full simulation does not separate concerns.
3. Solution: two modules.

Design problems--Tradeoffs and compromises

B. Devices with several independent characteristics

1. Failure to fully separate.

2. Solution: Module within module.

Design problems--Tradeoffs and compromises

C. Changeable characteristics that cannot be hidden

1. System generation parameters

1.1 Generation time

2.2 Load time

Design problems--Tradeoffs and compromises

D. Device dependent data to/from other modules

- Restricted interfaces

Design problems--Tradeoffs and compromises

E. Removable interconnections between devices

- Design upward compatible interface

Design problems--Tradeoffs and compromises

F. Device interdependence

- provide UE messages that hide the interdependence

Design problems--Tradeoffs and compromises

G. How to report an event

1. Specify both events and access programs.
2. Implement only what is used.

Design problems--Tradeoffs and compromises

H. Devices that need software supplied information

- Is this on the interface or hidden?
- Hidden, use other programs to get information.
- On interface, provide program to receive information.

Design problems--Tradeoffs and compromises

I. Correspondence between real and virtual devices

Bottom Line:

The basic definition of abstraction gives good guidelines even in hard design problems.

We can do a better job with a systematic procedure and a principle.