SECTION 2

ACCESS.bus

Base Protocol Specification

TABLE OF CONTENTS

ABSTRACT

ACCESS.bus provides a simple, uniform way to link a local computer to fourteen or more low-speed I/O devices such as keyboards, mouse, tablet, and 3-D tracker. ACCESS.bus features a one-hundred kilobits per second maximum data rate, hardware arbitration, dynamic reconfiguration, a mature capabilities grammar to support generic device drivers, and off-the-shelf, low-cost I^2C microcontroller technology.

This specification is offered as an open interface standard. Published revisions of this document must be approved by the ACCESS.bus Industry Group.

2.0 ACCESS.bus Protocol

2.1 General Description

Every device on the bus must be a microcontroller with I^2C interface and behave as either a master transmitter or a slave receiver, exclusively, as defined in the I^2C Specification. I^2C allows for master receiver and slave transmitter, but these modes are not used by the ACCESS.bus protocol.

The computer microcontroller acts as an interface converter and FIFO. Its primary job is to remove I^2C specific burden from the device driver as well as to take care of the computer I^2C address and checksum generation/checking. The computer microcontroller's secondary task is to monitor the "Hardware Signal" family of commands. When it receives one of these commands it asserts a signal in addition to passing the message to the computer. These signals provide hardware attention, reset, and halt capability from a standard keyboard.

A message transmits information between a device and the computer or between the computer and one or more devices. There is one exception: a device may attempt to reset other devices assigned to the same address by sending a Reset message to itself.

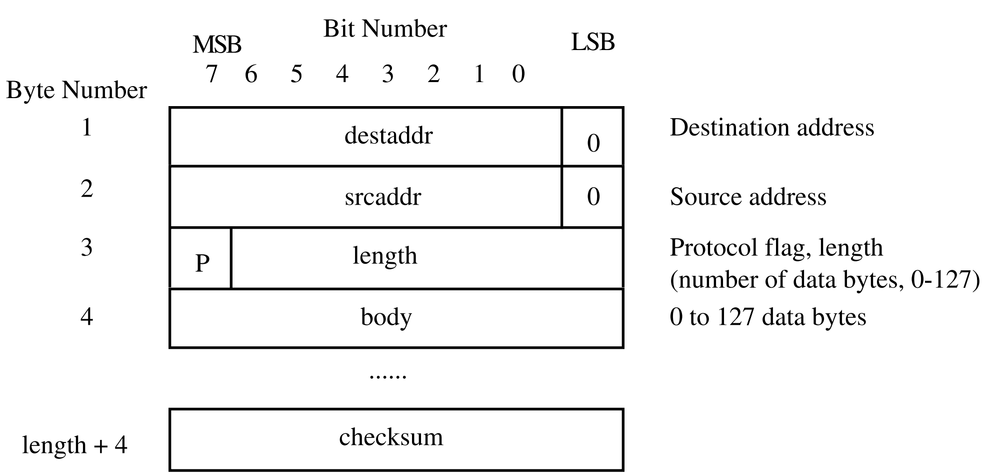
Initially, all devices respond to a default power up address. During the configuration process the computer assigns a unique address to every device on the bus. ACCESS.bus supports multiple like devices without switches or jumpers.

The bus supports dynamic reconfiguration while the system is operating. Connecting new devices shall not require powering down or rebooting the system before the new devices can be used ("hot plugging" shall be permitted).

2.2 Message Format

ACCESS.bus messages have the following format:

Figure 2.1: ACCESS.bus Message Format



Messages are either Device Data Stream (P=0) or control/status (P=1), as indicated by the protocol (P) flag. The minimum length of a message shall be four bytes. The maximum theoretical length of a message is 131 bytes (127 data bytes and four bytes for overhead); the maximum practical length is constrained by the transmission speed and the maximum time a device may hold the bus as master (see Timing Rules).

The message checksum shall be computed as the logical XOR of all previous bytes, including the message address. The checksum shall be computed such that the logical XOR of all previous bytes is equal to zero (0). A device or the computer shall only execute commands with a valid checksum.

ACCESS.bus data follows Big Endian bit order as defined by the I^2C specification. The most significant bits are always sent first.

The standard byte order for multibyte integers is Big Endian.

Figure 2.2: ACCESS.bus Message Example

S	6E	A	50	A	81	A	F1	A	4E	A	P

Where:

S	I^2C START signal
6E	destination address (device default)
A	acknowledge pulse from receiver
50	source address (computer)
81	Control/Status length 1
F1	Identification Request
4E	Checksum
P	I^2C STOP signal

2.3 Standard ACCESS.bus Protocol Messages

The ACCESS.bus protocol defines eight required interface messages that are summarized below. Parameters defined within the body of the message are listed in parenthesis.

Table 2.1: Protocol Messages

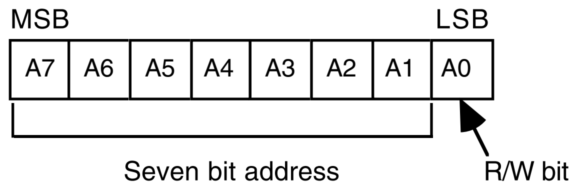
Computer-to-device Messages	Purpose
Reset()	Force device to power-up state and default I^2C address.
Identification Request()	Ask device for its "identification string."
Assign Address(ID strng, new addr)	Tell device with matching "identification string" to change its address to "new address."
Capabilities Request(offset)	Ask device to send the fragment of its capabilities information that starts at "offset."
Device-to-computer Messages	
Attention(status)	Inform computer that a device has finished its power-up/reset test and needs to be configured; "status" shall be the test result.
Identification Reply(ID string)	Reply to Identification Request with device's unique "identification string."
Capabilities Reply(offset, data frag)	Reply to Capabilities Request with "data fragment," a fragment of the device's capabilities string; the computer uses "offset" to reassemble fragments.
Interface Error ()	Invalid checksum or premature end of message detected

2.4 Addressing

ACCESS.bus addresses follow the I^2C address format defined in the I^2C specification. The seven most significant bits form a seven bit address (A7-A1), the LSB shall always be zero (0) indicating a master transmitter or write operation.

Note: Ten bit addresses are not permitted.

Figure 2.3 I²C Address Format



The following addresses are allocated for the ACCESS.bus (even numbers only):

50h	host computer address
6Eh	power up default address for all devices
52-6Ch	fourteen assign able device addresses

These addresses are reserved by Philips for microcontroller addressing, to prevent possible conflict with other I^2C components (for example: RAM, ROM, PLLs). An ACCESS.bus computer implementation must support the fourteen assignable device addresses noted above. A computer may assign devices to I^2C addresses outside this range after first insuring that these addresses are not in use by other equipment. Once assigned, these addresses must be treated the same as standard peripheral operating addresses.

The computer address shall never be changed unless the computer wants to simulate a device. This feature may be useful for debugging.

At power up or after a reset command, devices will respond to the default address (6Eh). No user data such as keystrokes or locator movement reports will be transmitted at the default address.

ACCESS.bus devices are not required to respond to the I^2C General Call address.

The least significant bit of the source address field which is not used for addressing (indicates R/W in destination address) is reserved for future protocol extension. Conforming devices shall transmit this bit as zero and ignore received messages in which this bit is one until such extensions, if any, are defined.

2.5 Identification

ACCESS.bus is a bus-topology network that uses unique identification strings to distinguish devices. These strings are structured as follows:

protocol revision:	1 byte ("A")
module revision:	7 bytes (e.g., "V1.0 ")
vendor name:	8 bytes (e.g., "DEC ")
module name:	8 bytes (e.g., "LK501 ")
device number:	32-bit signed integer

The protocol revision shall be a single byte used to identify the protocol implementation to the computer. This specification defines protocol revision "A" (41h). Any new protocol revisions shall be approved by the ACCESS.bus Industry Group.

The module revision, vendor name, and module name strings are left justified ASCII character strings padded with spaces. The content of these fields shall be determined by the device vendor.

The device number string shall be a 32-bit two's complement signed integer and may be either a random number (if negative) or a unique serial number (if positive).

It is important to note that the host software should never attempt to recognize a device based on the hardware ID string. Host software should always rely on the capabilities string fields for device identification.

2.5.1 Random versus Serial Device Numbers

Interactive devices may use either a random number or a (fixed) serial number in their Identification Reply messages. Unique serial numbers are more expensive to implement, but allow device identification and usage to be remembered between sessions when the system has been turned off or device hot plugged.

Non-interactive devices must use a (fixed) serial number since they cannot be distinguished by when they are first manipulated by the user.

The purpose of the 32-bit device number is to distinguish otherwise like devices with the same firmware. To aide ACCESS.bus management software, serial numbers are reported as positive (2's complement integers), while random numbers are always reported as negative.

If a pseudo random number is used in the Identification Reply message, it must be produced in a way that will help distinguish like devices.

Guideline: The number of clock cycles since power on at the time a command is received may be used as a pseudo random number. The natural dispersion of resonator frequencies is usually sufficient to separate otherwise identical devices.

A new pseudo random number must be generated if a reset command is received. The pseudo random number shall not change between identify reports unless an intervening power-up or reset command occurs.

2.6 Capabilities Information

Device capabilities is the set of information that describes the functional characteristics of an ACCESS.bus peripheral. The purpose of capabilities information is to allow software to recognize and use the features of bus devices without prior knowledge of their particular implementation. By having locator devices report their resolution, for example, generic software can be written to support a range of device resolutions. Capabilities information provides a level of device independence and modularity.

The structure of capabilities information is designed to be simple and compact for efficiency, but also extensible to support new devices without requiring changes to existing software or peripherals. These objectives are supported by making the structure hierarchical and representing capabilities information in a form that applications (and humans) can use directly. The capabilities information shall be an ASCII string constructed from a simple, readable grammar. The capabilities string for a locator might read as follows:

```
(
 prot(locator)
 type(mouse)
 model(VSXXX-AA)
 buttons(1(L)2(R)3(M))
 dim(2) rel res(200 inch) range(-127 127)
 d0(dname(X))
 d1(dname(Y))
)
```

Capabilities information shall be static. The information shall describe the potential operating modes and characteristics of a device. From the point of view of an ACCESS.bus peripheral, capabilities information can simply be a string of bytes which is transferred to the computer via the Capabilities Request and Capabilities Reply commands.

Capabilities information must always be sent in the following order:

```
prot()
type()
model()
```

These elements must always be the first three items in the capabilities string and they must occur within the first sixty-four (64) characters.

2.6.1 Capabilities String Syntax and Semantics

Capabilities information shall be an ASCII string constructed from the simple grammar as follows.

1. The terminal symbols of the grammar are STRING, TAG, WS, '(', and ')'.
2. WS is a sequence of one or more white space characters: SPACE, TAB, RETURN or LF.
3. A STRING is a sequence of one or more non-white space characters. All Capabilities data shall be represented as STRING: integers (123), floats (+3.0e8), strings (keyboard). Special characters may be included in STRINGs by escaping them as \xHH where HH represents two hexadecimal digits. SPACE, TAB, RETURN, LF, '(', ')', and '\' are special characters and must be escaped as \HH to include them in STRINGs.
4. A TAG is a STRING which is immediately followed by a '('.
5. '(' and ')' are open and close parenthesis used for grouping.

The grammar allows STRINGs to be formed into lists separated by white space, lists with tagged elements, and nested lists according to the following rules (BNF).

```
Capabilities ::= ( cap string )
cap string ::= STRING
```

cap string ::= cap string WS cap string
cap string ::= TAG cap string)
cap string ::= TAG cap string) cap string

Notes:

1. All 8-bits of characters are significant. This is to allow 8-bit multinational character sets such as ISO Latin-1 to be used.

2. Keyword comparisons are **not** case sensitive.

3. Capabilities are hierarchical. The meaning of a keyword within a tagged list depends on the tagged list in which it appears.

2.6.2 Standard Capabilities and Conventions

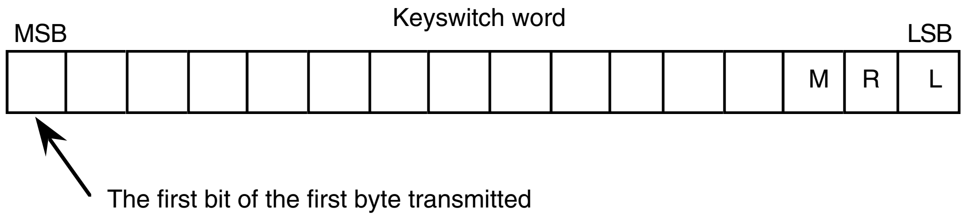
Certain keywords appearing at the top level of a capabilities string are defined to have standard meanings within the ACCESS.bus protocol as follows:

Keyword	Meaning
prot()	The "prot()" entry identifies the generic protocol or device type to system software. prot() is the device driver's view of a device. prot() defines the behavior of a device in terms of commands and responses in addition to those defined in the Base Protocol.
prot(keyb)	generic keyboard
prot(locator)	generic locator
prot(text)	generic text
prot(joystick)	generic joystick
type()	The "type()" entry is intended to identify the device type to the user in a recognizable form. type() is a user's view of a device. That is, a joystick, mouse, or whatever.
type(keyboard)	
type(mouse)	
type(tablet)	
type(trackball)	
type(barcode)	
model()	The "model()" entry is provided to present the full model name to the user if different from the module name. The "model()" entry is optional.

The following capability usage convention is recommended to maximize compatibility between hardware and software from different vendors.

> Tags used as keywords to identify device features should be no more than eight characters in length (only the first eight characters are significant). Characters A-Z, a-z, and 0-9 are assumed to be handled transparently on all systems.

Figure 2.4: Keyswitch Word



3. Device features or events that are represented by bit positions should be numbered sequentially starting with "1" for the least significant bit. For example, locator buttons within a 16 bit keyswitch word might be identified as "buttons(1(L)2(R)3(M))".

4. Features controlled by a numeric value parameter can be described by the feature name as a tag followed by the parameter range. If the range is from zero to maximum, the zero minimum value can be omitted. Keyclick volume ranging from 0 (off) to 7 (maximum) can be described as "click(7)" for example. It is suggested that these values not be normalized. By not normalizing it is possible to preserve the full resolution of the device, thus allowing the computer to determine the smallest meaningful volume increment.

Specific additional capabilities are defined in generic device protocol specifications. Generic specifications for keyboards, locators, and text devices have been developed to date.

2.7 Configuration Process

The configuration process shall be used to detect the devices that are present on the bus, assign each device a unique address, and connect devices to the appropriate software driver. Configuration shall occur at system start-up, or at any time when the computer detects the addition or removal of a device.

2.7.1 Power-up/Reset Phase

When reset or powered-up, a device shall always revert to the default address and send an Attention message to alert the computer to its presence. At system start-up or reinitialization, the computer shall send a Reset message to all I^2C addresses in the ACCESS.bus device address range (fourteen messages) to insure that all devices on the bus respond at the power up default address.

If a device is reconfigured by the user, it shall revert to the default address and send an attention message while keeping the same identification string and device number.

2.7.2 Identification Phase

To begin address assignment, the computer sends an Identification Request message at the device default address. Every device at this address must then respond with an Identification Reply message. As each device sends its message, the I^2C arbitration mechanism automatically separates the messages based on the identification strings. The computer can then assign an address to each device by including the matching identification string in the Assign Address message. A device that receives this message and finds a complete match with the identification string moves its device address to the new assigned value. As soon as a device has a unique address, it shall be allowed to send data to the computer.

The I^2C physical bus protocol allows multiple devices on the bus at the same time, if those devices are transmitting exactly the same message. In the rare event that two like devices report the same random number or are mistakenly assigned to the same address, each interactive device transmits a Reset message to its assigned address immediately prior to sending its first data message after being assigned a new address. The self-addressed Reset message forces other devices at the same address back to the power-up default address, as if they had just been hot-plugged. The message guarantees that each device has a unique address, but not until the device is actually used. The pseudo random number (or serial number, if available) distinguishes devices at identification time before they are used, allowing the computer to inventory which devices are present.

2.7.3 Capabilities Phase

After assigning a unique address to a device, the computer retrieves the device's capabilities string as a series of fragments using the Capabilities Request and Capabilities Reply messages. The computer then parses the capabilities string to choose the appropriate application driver for the device. The parsed string shall also be made available to application programs using the device.

2.7.4 Normal Operation

During normal operation, the computer will periodically request inactive devices to identify themselves. If a device is found to be missing, or a new device appears by sending an Attention message at the default address, the computer will send an Identification Request message to each device address previously recorded as in use (up to fourteen messages) to confirm which devices are still present. The computer will also send a Reset message to each device address previously recorded as not in use. The computer then begins the address assignment process by sending an Identification message to the default address and assigning each device that responds to an unused device address.

2.8 Timing Rules

2.8.1 Bus Timing

The ACCESS.bus is designed primarily for interactive I/O devices. Interactive devices are defined as those that only generate device data stream bus traffic in response to being manipulated by the user. To insure reasonable interactive performance, ACCESS.bus devices and computers must conform to the following rules:

1. Interactive devices shall not occupy the bus as bus master for more than eight milliseconds (8ms) at a time. This is to insure other input devices have a chance to update the screen every frame time (16.6ms at 60Hz) which is necessary for smooth animation. Eight milliseconds (8ms) limits the maximum packet size to eighty characters or less at 100Kbps.

Interactive devices must allow at least fifty microseconds (50µsec) between releasing bus mastership at the end of a message and requesting to become bus master again. This is to give other interactive devices a chance to access the bus without arbitration. If the bus is not otherwise busy, interactive devices may utilize 80% or more of the bus bandwidth.

ACCESS.bus computers shall conform to this rule for interactive devices.

2. Non-interactive devices shall not occupy the bus as bus master for more than five milliseconds (5ms) at a time. Non-interactive devices must allow at least twelve milliseconds (12ms) between releasing bus mastership at the end of a message and requesting to become bus master again. This is to reduce the possibility of non-interactive devices compromising interactive performance. Non-interactive devices are restricted to using no more than 30% of the bus bandwidth. Five milliseconds (5ms) limits the maximum packet size to fifty characters or less at 100Kbps.

3. ACCESS.bus interfaces shall not hold SCL low for more than two milliseconds (2ms). A watchdog timer or other provision shall be made to release SCL before the two millisecond (2ms) limit is reached.

2.8.2 Response Timing

Time limits for certain commands to execute are specified so that the computer can determine when all the devices present have had sufficient time to respond (time out). The following limits apply:

1. Devices shall complete the Reset command within 250ms. This is believed to be long enough for basic power up self test (from stable power) without causing excessive delays.

2. Devices shall respond to all other commands that require a response within forty milliseconds (40ms).

3. If a command can be responded to by more than one device, the time limit shall be extended to forty milliseconds (40ms) since the last device that responded.

2.9 Exception Handling

The following general requirements and recommendations are defined for handling I^2C and ACCESS.bus exception conditions. Additional requirements may be defined by the individual device protocol specifications.

1. Arbitration Loss - If a device detects that it has lost arbitration, the device shall cease transmitting and try again to become bus master to re-send the message.

2. Negative Acknowledge - If a device transmits a byte which is negatively acknowledged (no receiver, or rejected for some reason), or if the bus times out, the

device shall abort the transfer immediately by generating a Stop condition. No retry is required. A small number of retries may be attempted.

3. Checksum Error - If a device detects a checksum error in a received message, the message data shall be ignored. It is recommended the device signal an Interface Error to the computer.

4. Premature STOP - If a device detects a premature I^2C STOP signal before the end of a message is reached, the message data shall be ignored. It is recommended the device signal an Interface Error to the computer.

5. If the computer receives an Interface Error message from a device, or detects a checksum error or premature I^2C STOP condition, it is suggested the computer log the error and re-issue its most recent request to that device. A large number of interface errors may indicate a software error or faulty hardware configuration. In this case, it is suggested the computer notify the user and attempt to restart the configuration process.

6. Unrecognized Commands and Parameters - If a device receives a command with valid checksum but does not recognize the command op-code, the entire command shall be ignored. If a device receives a command with valid checksum but does not recognize the value of a required parameter, the entire command shall be ignored. If a device receives a command with valid checksum that includes additional data beyond that expected, it is recommended the additional data be ignored, however, the device may execute the command.

2.10 Detailed Command/Message Descriptions

2.10.1 Command Coding

For the purpose of illustrating message encoding, the following notation is used:

```
P        = 1-bit protocol flag:0=Data Stream;
           1=control/status message
LLLLLLL  = 7-bit length field Encodes values 0-127
ddddddd  = 7-bit destination I2C address
sssssss  = 7-bit source I2C address
xxxxxxx  = 6-bit or 7-bit message op-code
cccccccc = 8-bit checksum
```

Numeric values are always transmitted MSB first.

ACCESS.bus messages are either Device Data Stream (P=0), or Control /Status (P=1). Data Stream messages always refer to the application part of a device, as opposed to the interface part. The coding of Data Stream messages is dependent on the device. They are not pre-defined or restricted by the ACCESS.bus protocol.

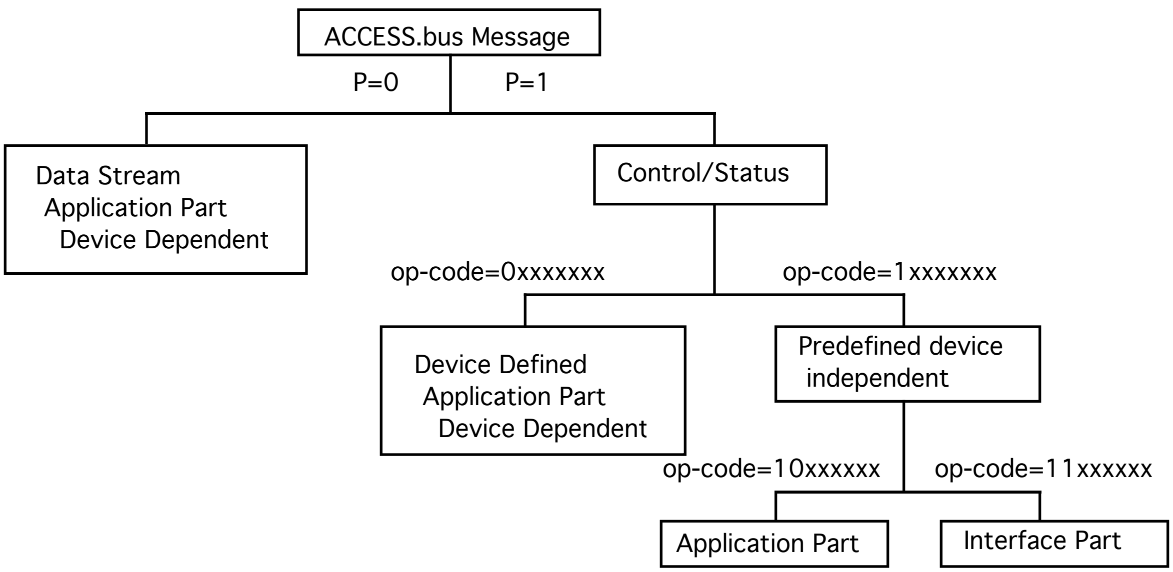
```
   Data Stream Msg                  Control/Status Msg

         ddddddd0                        ddddddd0
         sssssss0                        sssssss0
         PLLLLLLL    P=0                 PLLLLLLL    P=1
         |                               |
```

```
body[0-127]             xxxxxxxx    op-code
|                       body[0-126]
cccccccc                |
                        cccccccc
```

The first data byte of Control/Status messages (P=1) shall always be an op-code. Control/Status messages may refer to either the application part or the interface part of a device. Pre-defined Control/Status op-codes (messages) are used to initialize and configure ACCESS.bus devices. A range of Control/Status op-codes are reserved for private use by devices and will not be pre-defined as part of the ACCESS.bus protocol.

The breakdown of ACCESS.bus message types is shown in the following diagram:

Figure 2.5: ACCESS.bus Message Types Breakdown



2.10.2 Device Data Stream Message

These messages are usually used for the bulk of the device's data. The intent is that the most common messages (keypress reports, locator movements, etc.) should be easiest to send and have the lowest overhead. Device Data Stream messages are distinguished by the Protocol bit being zero. The body of these messages are defined by the individual device protocol specifications.

Devices are not permitted to send this message at the power on Default Address. After a successful Assign Address command, devices shall send a self addressed reset message immediately prior to transmitting user data for the first time (see Section 2.7 - Configuration Process).

Three standard device protocols are defined at this time: (1) keyboard (down/up key transitions); (2) locator device (relative or absolute position, i.e. mouse, tablet,...); (3) text device (ISO Latin-1 text, i.e. card reader). Refer to the appropriate device protocol specifications.

2.10.3 Device Defined Control/Status (C/S) Messages

Out-of-band control or status information shall be sent with the protocol flag set to one (P=1). Op-codes in the range 00h to 7Fh are reserved for device defined peripheral messages (application part). Refer to the individual device protocol specifications for details.

2.10.4 Pre-defined C/S Messages, Interface Part

Interface Part Control/Status messages are used to control the ACCESS.bus itself.

2.10.4.1 Reset

A Reset message shall be used to instruct the addressed device(s) to reset to the power up state.

Format:

```
ddddddd0
sssssss0
10000001    (81h, P=1, length=1)
|
11110000    (F0h, Reset op-code)
|
cccccccc
```

This command is intended to be completely equivalent to power on reset including changing of device address to the Default Address, power on self testing, and transmission of an Attention message at the Default Address.

Notes:

1. The device shall ignore all commands from the computer until it has completed power-up processing and transmitted a successful Attention message. This is the only command which may cause a device to temporarily ignore bus messages.

2. A device must wait at least eight milliseconds (8ms) after its I^2C hardware is reset before transmitting to insure its I^2C hardware has synchronized with any message frame in progress.

3. The Reset command must be completed within 250ms (excluding the time spent waiting to become bus master).

4. Because the ACCESS.bus does not have a General Call or Broadcast address, the ACCESS.bus shall be reset by issuing Reset commands to all fourteen standard ACCESS.bus device addresses and any other addresses that the computer has explicitly assigned.

2.10.4.2 Attention

An Attention message shall be sent by a device to the computer. This message reports the health of the electronics and firmware after power up or reset. The message serves two purposes:

1. To notify the computer a device needs attention after power up or reset.

2. To confirm basic device function during manufacturing.

Format:

```
dddddddd0
sssssss0
1LLLLLLL    (P=1, length=2-32)
|
11100000    (E0h, Attention op-code)
status      (0=success, non-zero=failed)
data        (0-30 bytes)
|
cccccccc
```

Except for the Status byte, the body of the Attention message is device dependent. The remainder of the body may be used for any device-specific test related or manufacturing information. If possible, failure data should be human readable since it may be displayed to the user via a dialog box if a device fails.

An Attention message with a status byte of zero shall indicate that the device is capable of responding to the other ACCESS.bus commands and of operating in some useful manner. Further testing, if needed, shall be device-dependent and is left up to the application program or driver.

Status not equal to zero indicates that the device has detected some error condition that prevents normal operation. After reporting non-zero status, the device should attempt to continue testing the error condition for the possibility it will be corrected. If and when the device detects all such error conditions have been corrected, the device shall transmit a "successful" Attention message to indicate it is now ready for normal operation just like a device that has been hot plugged.

For example, if a keyboard detects a key down error during power-up, it should continue scanning until all keys have been released, and then report success.

Notes:

1. This message must be transmitted within 250ms after a reset command or stable power is applied (excluding time spent waiting to become bus master).

2. A device must wait at least eight milliseconds (8ms) after its I^2C hardware is reset before transmitting to insure its I^2C hardware has synchronized with any message frame in progress.

3. The device must ignore all commands from the computer until it has completed power-up processing and transmitted a successful Attention message. Devices that fail must not respond to any commands.

4. Because Attention is only generated after power on or reset, it will always be transmitted at the Default Address. A status equal to zero shall indicate to the computer that a device is present on the bus and awaiting configuration.

5. The computer should take no action to configure or use devices that report non-zero status since the device is not capable of normal operation. The non-zero status is intended for manufacturing testing. During manufacture there is no need to identify which device has failed. The bytes following the status byte are used for manufacturing test information.

2.10.4.3 Identification Request

An Identification Request message shall instruct the addressed device(s) to send a complete identification report (see Section 2.10.4.4 - Identification Reply).

Format:

```
dddddddd0
sssssss0
10000001    (81h, P=1, length=1)
|
11110001    (F1h, Identification Request op-code)
|
cccccccc
```

The Identification Request message will: (1) Cause all devices at the Default Address to send a unique identification string so that the computer can assign each device to a distinct address; (2) Check each device to determine if the device is still connected to the ACCESS.bus and responding at the assigned address.

2.10.4.4 Identification Reply

The Identification Reply message shall be issued by a device in reply to an Identification Request message. The reply shall consist of a string that identifies the device hardware. A four byte pseudo random number (or serial number if available) shall be included in order to distinguish like hardware devices.

Format:

```
ddddddd0
sssssss0
10011101          (P=1, length=29)
|
11100001          (E1h, Identification Reply op-code)
protocol revison  (1 byte, "A")
module revision   (7 bytes, e.g., "V1.0   ")
vendor name       (8 bytes, e.g., "DEC     ")
module name       (8 bytes, e.g., "LK501   ")
device number     (32 bit signed integer)
|
cccccccc
```

The module revision, vendor name, and module name shall be left justified ASCII strings padded with the space character (20h). See "Random versus Serial Device Numbers" for a description of the device number.

2.10.4.5 Assign Address

An Assign Address message shall be issued by the computer to instruct the addressed device(s) with matching identification strings to move to the specified device address.

Format:

```
ddddddd0
sssssss0
10011110          (P=1, length=30)

11110010          (F2h, Assign Address op-code)
protocol revison  (1 byte,  "A")
module revision   (7 bytes, e.g., "V1.0   ")
vendor name       (8 bytes, e.g., "DEC     ")
module name       (8 bytes, e.g., "LK501   ")
device number     (32 bit signed integer)
new I2C address   (1 byte)

cccccccc
```

When a device receives this command and finds a complete match in the protocol revision, module revision, vendor name, module name, and device number field it moves its address to the new assigned value. If the device is successfully assigned a new address, it shall be able to respond to messages at the new address immediately, that is, on the next valid I^2C message frame.

If the identify information does not match that of the receiving device, the entire message shall be ignored.

Devices shall not transmit user data while at the Default Address. If the Assign Address is successful, the device shall send a self addressed reset command on its new address immediately prior to sending its next data report. Once a new address is assigned, the device may begin transmitting at the new address without waiting for the configuration process to complete.

2.10.4.6 Capabilities Request

A Capabilities Request message shall be issued by the computer to a device to instruct the addressed device to reply with a Capabilities Reply. The Capabilities Reply shall contain data starting at "offset".

Format:

```
ddddddd0
sssssss0
10000011    (P=1, length=3)
11110011    (F3h, Capabilities Request op-code)
offset      (16-bit unsigned integer)
cccccccc
```

The Capabilities Request and Capabilities Reply messages form a protocol for transferring an arbitrary byte-string from the device to the computer, via a series of fragments. "Offset" shall be the index (from 0) into this string. To simplify the device's implementation of this protocol, "offset" shall be restricted to three values:

"send first" zero, indicating the computer wants to start over at the beginning;

"send again" the offset from the most recently transmitted Capabilities Request, indicating the computer did not receive a response and wants a retransmit;

"send next" the offset from the most recently received Capabilities Reply plus the number of bytes in the message fragment. (new offset = old offset + fragment length) (fragment length = message length - 3).

With these restrictions the computer can make three requests: start over, send current, and send next. See Capabilities Reply message for further details.

2.10.4.7 Capabilities Reply

A Capabilities Reply shall be used to reply to a Capabilities Request message with a fragment of data starting at "offset".

Format:

```
ddddddd0
sssssss0
100LLLLL    (P=1, length=3-35)
|
11100011    (E3h, Capabilities Reply op-code)
offset      (16-bit unsigned integer)
data        (0-32 bytes)
|
cccccccc
```

The protocol is designed to be simple for the device to implement: the device designer is free to allow the device to choose the most convenient fragment size from one message to the next.

The only state information the device should need to maintain is the current offset and length of the most recently transmitted fragment.

On receiving a Capabilities Request message, the device shall examine the "offset" field:

- If equal to zero, the device shall set the current offset to zero and send the fragment from offset zero (0).
- If equal to the current offset, the device shall re-send the fragment from the current offset.
- If equal to the "current offset" + "fragment length", the device shall update the current offset (current offset := current offset +

fragment length) and then look up (or calculates) the next fragment to send and sends it.

- If the device has reached end-of-string, it shall send a fragment with the next offset but zero data bytes. This will indicate an end of string.

- Otherwise, the device shall set the "current offset" to zero and send the fragment from offset 0.

2.10.4.8 Interface Error

An Interface Error command shall be used to inform the sender that the last message addressed to this device was not received successfully; either a checksum error or premature end of message (I^2C STOP condition) was detected.

Format:

```
dddddddd0
sssssss0
10000001    (81h, P=1, length=1)
|
11100100    (E4h, Interface Error op-code)
|
cccccccc
```

2.10.4.9 Resource Request (Optional)

A Resource Request shall be sent by a device to request a resource from the computer.

Format:

```
ddddddd0
sssssss0
100xxxxx    (P=1, length=n)
|
11100101    (E5h, Resource Request op-code)
res code    (resource designator)
data        (optional data)
|
cccccccc
```

The following resource codes are supported:

1. Request I^2C address for private use.
   Data byte 1, if specified, shall be the desired I^2C address.

2. Relinquish I^2C address.
   Data byte 1 shall be the I^2C address to be relinquished.

3. Request time and date.
   No additional data provided.

2.10.4.10 Resource Grant (Optional)

The Resource Grant command shall be sent by the computer to a device to indicate that a requested resource has been granted to the requesting device.

Format:

```
dddddd0
sssssss0
100xxxxx    (P=1, length=n)
|
11110100    (F4h, Resource Grant op-code)
res code    (resource designator)
status      (0=success, 1=unsupported,
             2=failed - try again later, 4=failed)
data        (optional data)
|
cccccccc
```

The following resource codes are supported:

1. I^2C address granted for private use.
   Data byte 2 shall be the I^2C address.

2. Response to relinquish I^2C address.
   Data byte 2 shall be the I^2C address relinquished.

3. Current time and date.
   Data bytes 2,3 = year
   Data byte 4 = month
   Data byte 5 = day of month
   Data byte 6 = hour
   Data byte 7 = minute
   Data byte 8 = second
   Data byte 9 = .01 second

2.10.4.11 Vendor Reserved Commands (Optional, Interface Part)

A small range of op-codes are reserved for vendors to invoke private functions for testing, alignment, or set-up at manufacturing time. These commands might be used to down load an EEPROM with a serial number for example. These commands are device dependent and should not be used during normal ACCESS.bus operation. If these commands invoke any private modes, it is recommended that such modes be exited by the Reset command (op-code F0h).

Table 2.2: Vendor Reserved Op-codes and Functions

Op-code (hex)	Function
C0	Reserved for vendor specific use
C1	Reserved for vendor specific use
C2	Reserved for vendor specific use
C3	Load information to device (NVM)
C4	Read or report device serial number

C5	Switch to low power mode
C6	Invoke manufacturing test
C7	Diagnostic request (like capabilities request)
C8	Diagnostic reply (like capabilities reply)

2.10.5 Pre-defined C/S Messages, Application Part

2.10.5.1 Application Hardware Signal

The Application Hardware Signal shall be sent by a device to the computer. It shall be used to instruct the computer I^2C controller to generate a high priority hardware signal.

Format:

```
dddddddd0
sssssss0
10000010    (P=1, length=2)
|
10100000    (A0h, Application Hardware Signal op-code)
sigcode     (signal code 1=reset, 2=halt, 3=attention)
|
cccccccc
```

The action taken by the computer on receiving this messages shall be dependent on the computer. The intent is that the computer's I^2C controller physically interrupt the computer.

Table 3: Application Hardware Signals

Number	Name	Purpose
1	Reset	Attempt soft reset of host computer. If no response within pre-set time limit, activate hardware reset of host computer
2	Halt	Primarily a debugging tool; could cause exit to a ROM debugger
3	Attention	Gain the attention of the computer

This command is required on the host computer, but optional for devices.

2.10.5.2 Application Test

An Application Test command shall instruct the addressed device to reply with the results of a self-test specific to the device.

Format:

```
ddddddd0
sssssss0
10000001    (P=1, length=1)
|
10110001    (B1h, Application Test op-code)
|
cccccccc
```

This command shall be used by the Application driver to test the Application Part of the device. When the command is issued, the device has already been configured so the basic ACCESS.bus functions are assumed to be working.

Unlike power up testing, the device must respond promptly and may not continue testing until the test succeeds; the device may not ignore other bus commands while testing; and the testing shall not alter the application state.
If no application specific testing is required, it is recommended that the device report the previous test results gathered during power up testing.

If useful in a particular application, the device may return a new test report each time the Application Test command is received. The device's Application driver may then issue Application Test commands until the device signals success.

2.10.5.3 Application Test Reply

An Application Test Reply shall reply to an Application Test message with a self-test report specific to the device.

Format:

```
ddddddd0
sssssss0
```

```
1LLLLLLL    (P=1, length=2-32)
|
10100001    (A1h, Application Test Reply op-code)
status      (0=passed, non-zero=failure)
body        (0-30 bytes of additional test
             information)
|
cccccccc
```

The only restriction on the message is that the first byte of the body (status) be zero if the test succeeded, and non-zero otherwise.

The computer may respond to a failed test report by either ignoring the message (and the device), or printing the body of the message, perhaps in both hexadecimal and ASCII (if printable).

APPENDIX A

SUMMARY OF STANDARD ACCESS.BUS INTERFACE OP-CODES

Op-code (hex)	Function
F0	Reset
F1	Identification Request
F2	Assign Address
F3	Capabilities Request
E0	Attention
E1	Identification Reply
E3	Capabilities Reply
E4	Interface Error
C0	Reserved for vendor specific use
C1	Reserved for vendor specific use
C2	Reserved for vendor specific use
C3	Load information to device (NVM)
C4	Read device serial number
C5	Switch to low per mode
C6	Invoke manufacturing test
C7	Diagnostic request (like capabilities request)
C8	Diagnostic reply (like capabilities reply)
B1	Application Test
A0	Application Hardware Signal
A1	Application Test Reply