This is a summary of the most important conventions for use of fault codes in the I2C/SMBus stack.

A "Fault" is not always an "Error"

Not all fault reports imply errors; "page faults" should be a familiar example. Software often retries idempotent operations after transient faults. There may be fancier recovery schemes that are appropriate in some cases, such as re-initializing (and maybe resetting). After such recovery, triggered by a fault report, there is no error.

In a similar way, sometimes a "fault" code just reports one defined result for an operation ... it doesn't indicate that anything is wrong at all, just that the outcome wasn't on the "golden path".

In short, your I2C driver code may need to know these codes in order to respond correctly. Other code may need to rely on YOUR code reporting the right fault code, so that it can (in turn) behave correctly.

I2C and SMBus fault codes

These are returned as negative numbers from most calls, with zero or some positive number indicating a non-fault return. The specific numbers associated with these symbols differ between architectures, though most Linux systems use <asm-generic/errno*.h> numbering.

Note that the descriptions here are not exhaustive. There are other codes that may be returned, and other cases where these codes should be returned. However, drivers should not return other codes for these cases (unless the hardware doesn't provide unique fault reports).

Also, codes returned by adapter probe methods follow rules which are specific to their host bus (such as PCI, or the platform bus).

EAGAIN

> Returned by I2C adapters when they lose arbitration in master transmit mode: some other master was transmitting different data at the same time.
>
> Also returned when trying to invoke an I2C operation in an atomic context, when some task is already using that I2C bus to execute some other operation.

EBADMSG

> Returned by SMBus logic when an invalid Packet Error Code byte is received. This code is a CRC covering all bytes in the transaction, and is sent before the terminating STOP. This fault is only reported on read transactions; the SMBus slave may have a way to report PEC mismatches on writes from the host. Note that even if PECs are in use, you should not rely on these as the only way to detect incorrect data transfers.

EBUSY

Returned by SMBus adapters when the bus was busy for longer than allowed. This usually indicates some device (maybe the SMBus adapter) needs some fault recovery (such as resetting), or that the reset was attempted but failed.

EINVAL

This rather vague error means an invalid parameter has been detected before any I/O operation was started. Use a more specific fault code when you can.

One example would be a driver trying an SMBus Block Write with block size outside the range of 1-32 bytes.

EIO

This rather vague error means something went wrong when performing an I/O operation. Use a more specific fault code when you can.

ENODEV

Returned by driver probe() methods. This is a bit more specific than ENXIO, implying the problem isn't with the address, but with the device found there. Driver probes may verify the device returns *correct* responses, and return this as appropriate. (The driver core will warn about probe faults other than ENXIO and ENODEV.)

ENOMEM

Returned by any component that can't allocate memory when it needs to do so.

ENXIO

Returned by I2C adapters to indicate that the address phase of a transfer didn't get an ACK. While it might just mean an I2C device was temporarily not responding, usually it means there's nothing listening at that address.

Returned by driver probe() methods to indicate that they found no device to bind to. (ENODEV may also be used.)

EOPNOTSUPP

Returned by an adapter when asked to perform an operation that it doesn't, or can't, support.

For example, this would be returned when an adapter that doesn't support SMBus block transfers is asked to execute one. In that case, the driver making that request should have verified that functionality was supported before it made that block transfer request.

Similarly, if an I2C adapter can't execute all legal I2C messages, it should return this when asked to perform a transaction it can't. (These limitations can't be seen in the adapter's functionality mask, since the assumption is that if an adapter supports I2C it supports all of I2C.)

EPROTO

Returned when slave does not conform to the relevant I2C or SMBus (or chip-specific) protocol specifications. One case is when the length of an SMBus block data response (from the SMBus slave) is outside the range 1-32 bytes.

ETIMEDOUT

This is returned by drivers when an operation took too much time, and was aborted before it completed.

SMBus adapters may return it when an operation took more time than allowed by the SMBus specification; for example, when a slave stretches clocks too far. I2C has no such timeouts, but it's normal for I2C adapters to impose some arbitrary limits (much longer than SMBus!) too.