I2CMaster with 2 slaves: Robust data transfer

The attached code contains 3 .ino files. One for the master and two for the slaves.
This is the example code for the MersAG meeting

Brief description

The master receives data from the LSlave (low slave) and sends and receives data to the HSlave (high slave) the H&L names are just convenience,

Before writing or reading from the slave Arduinos the master performs some checks, crash prevention and corrective actions:

A variable LslavePresent and HslavePresent track the status of each device. If either are 0 then code in the program can use default values or not attempt to read the devices to avoid lockups when trying to read devices that dont exist or have failed connections.

bool checkI2CSlave(uint8_t address, bool &presentFlag){}

checkI2CSlave() returns true or false when sent the slave address and the referenced presentFlagfor that device. The referenced value for both HSlavePresent and LSlavePresent means that the actual values of these variables are updated, not the copy value created by the function which would normally be the case without the '&' before the argument name.

1. Transmission is initiated to the address passed to the function and then ended with nothing is read or written.The endTransmission returns values to indicate the status of the trial transmission
    a. 0: success.
    b. 1: data too long to fit in transmit buffer.
    c. 2: received NACK on transmit of address.
    d. 3: received NACK on transmit of data.
    e. 4: other error.
    f. 5: timeout
2. At this point we are only concerned that 0 is returned indicating a successful transmission. There is an Arduino there and it is connected. We could then use other codes to further establish recovery actions but this is beyond the scope of this example.
3. We can send a message saying the connection is present and then reset the presentFlag to stop flooding the Serial Monitor
4. If something other than '0' is returned the connection has failed. We send a message again and we call i2cBusRecover() in an attempt to restore the connection for the next loop iteration.

5. If it can’t recover it wont keep trying as the loss of connection is due to something else.

i2cBusRecover()

If the device connection fails it can fail mid process due to power loss, interference or loss of connection. This can cause the clock to stall awaiting acknowledgement from the now not present slave.

1. Set the SCL pin (A5 on Nano, Uno and siar) to output taking control of the clock line
2. Force the pin high then low
3. As each unit of transmission is 9 bits, doing this 9 times will reset the whole transmission irrespective of where in that transmission it crashed.
4. It then gives pin control back to Wire (sets it as INPUT_PULLUP) and delays 50 ms to settle down.

I2cLow() and i2cHigh()

These are called when the program wants to read or write to the particular slave device

1. It sends its address and the current slavepresent status to the checkI2CSlave function.
2. If it returns false (note the ! at the front of the function call reversing the true/false) the return statement immediately returns out of the function without further execution. No attempted read or write to a device that is faulty or absent.
3. If it has a connection it calls the number of bytes requested from that address
4. If there’s no data there it returns out of the function immediately
5. If there is it gets the last data written
6. It then fishes out the buffer so that if the salve has written many times to the buffer since the last read these are all cleared out. It is still possible that the buffer can overflow, but its now much less likely.
7. Once read the program can process the data it has just received.

If the device has become disconnected the program can detect this by the state of the slavePresent flag and take alternative action.