# Major changes

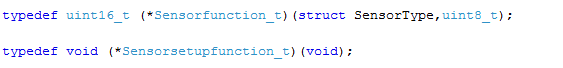
I’ve moved everything from header files into header/source file pairs

This makes the program easier to maintain and reuse (Some of the code needs to be reused for the diagnostics program and the CAR firmware I’m working on)

The main consequence of this is that the code is more segmented. Previously every file could see every other file.

The main way for files to interact now is through “Function Pointers” which we setup through type definitions. Look up “Function Pointer” if you’re unfamiliar with the term

Examples

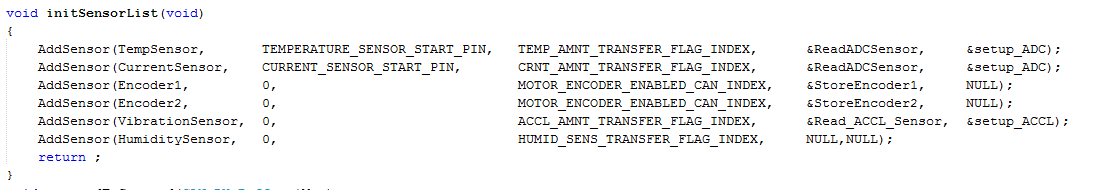


More info over page

## Add new sensor

I’ve tried to simplify this, so that you don’t need to edit as many files

In the file “main.c”, in the function “initSensorList”; add the “setup” and the “read” functions

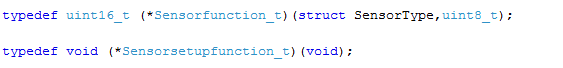


“SetupSensorListCODParameters ()” will call the setup function whenever required. The setup function should edit the “XXX\_AMOUNT\_TRANSFER\_FLAG\_INDEX” to zero if the sensor is not detected. “SetupSensorListCODParameters” will read this index after calling the setup function.

As a consequence you no longer have to manually call the setup function

If no setup function is required, then leave the value as null.

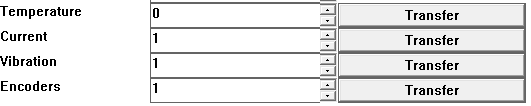
The functions must fit these definitions



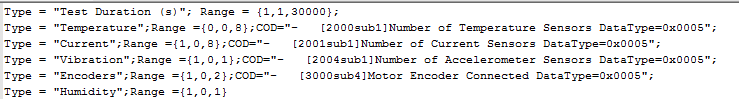
The setup function is void return with void arguments

The main function returns a 16 bit unsigned integer, must accept a “SensorType” and unsigned integer argument (Which specifies the output mode – RAM, CAN Bus etc.)

The “Transfer flag” is now removed from the code. The diagnostics program will have this build into the new “PDO Protocol”



Just click the transfer button; new sensors can be added in the config file “1. Configs/Condition Monitor.txt”. The “COD” index is only required for “set sensors”. The transfer flag will work even without it.



Then restart the program

You still need to define the amount of each sensor in the CM Object Dictionary

The process hasn’t changed, just add the new values in “ObjectDictionary.h/c”

# SPI communication

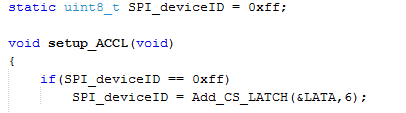
You no longer have to edit the “SPI Interface.h” file

You now utilise the chip select functionality with these three functions



## “Add\_CS\_LATCH”

* Returns a device ID. You can use this to set the CS High/Low. Save into your file as a static global value
* You provide the Latch register (i.e. &LATA). The program will calculate the TRIS register to set the data direction for you



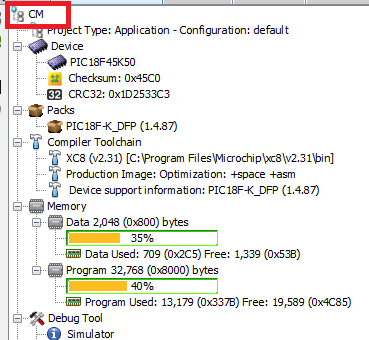
## “SET\_CS\_LOW” and “SET\_CS\_HIGH”

You have to call manually

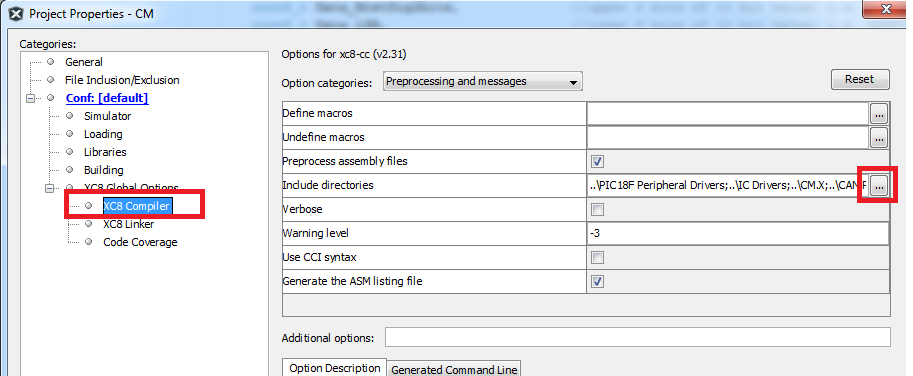


# Add new include directories

Open the project properties window



Click xc8 compiler and then the browser under “include directories”



Set up as so

