What is the power of Scripting languages (python) in Embedded world?

The mother language of the embedded world is C. It is an undeniable fact, but the problem with c or c++, they are old and dusky, and sometimes it is tough to do smart jobs. I found myself scripting languages can fix this issue to avoid repetition of the code and syntax errors.

Let’s say one such smart job that I picked is getting sensor register address and bit position, for small and medium sensors or controllers it looks easy to write one-time code in c. let's imagine there are tons of registers, and they have different bit sizes (preferably 8bit or 32bit).

I had recently written application-level firmware for STM LSM6DS3 Accelero - Gyro sensors during the firmware writing it became a mess for me to figure out 95 registers address and each register having 8 bits. Most of the time, I felt lazy to copy 95 register addresses and bit positions. I was to copy only the registers which I needed in the applications. As the application size and utilization keep increasing it became a mess to port the registers and register bits manually. In the end, I had to come to some smart way.

1. I just downloaded the data sheet of the sensor as a pdf
2. Imported Register table to Excel sheet ( MsExcel: Data->GetData->FromFile->FromPDF-> Select Table)
3. Step two can be also done by directly connecting Excel to the web ( Data->enter URL -> Select Table)
4. Once the table is imported to Excel modify it according to your requirement ( delete or add data columns ex: read status, default values of reg whatever)
5. Save the file in .xlsx format if you want to import more tables and complex data otherwise, you can save it as a .csv file)
6. Install Pandas or Excel and write an equivalent library.
7. Parse Excel file data into a data frame using pandas
8. Clean the data like .. white spaces or newlines
9. open header file using within python (with open('lsm6ds3.h','w') as file ) in writing mode.
10. Identify common patterns of bit blank like reserved or read-only, or predefined value ( in this particular example the table has value 0 is reserved and 1 is read so by those bit positions)
11. keep writing the c syntax in the file.write using the c syntax
12. Iterate dataframe register by register using dataframe.iterrows()
13. write each register name and address with the corresponding register address using the string format in Python ( file.write(f'#define {registerName} 0x{RegisterAddress}') )
14. write Even Register bits with position value and shifting

(example: file.write(f'#define {bit7}\_pos {pos} \n')

                        file.write(f'#define {bit7} (uint8\_t)(1UL << {bit7}\_pos) \n') )

1. In the end, you do not have to close the file, with a statement can close the file automatically.

The above method is one of the simple applications of the scripting language in embedded C,but let me give you an idea of how it saved my life. Through the above method, the script is written with 2050 lines of c code ..... I am very happy with it because it is nightmare for me to write such code manually within half an hour without any compilation error, but python did a great job for me.

A picture containing text, screenshot, number, font

Description automatically generatedA screenshot of a computer

Description automatically generated with low confidenceA picture containing text, electronics, screenshot, display

Description automatically generatedA screen shot of a computer program

Description automatically generated with low confidenceA screenshot of a computer program

Description automatically generated with medium confidence