**Integration of MATLAB-LSTM Model into foxBMS Firmware and Hex File Generation**

### ✅ Overview

This document outlines all the key modifications and steps needed to successfully integrate a MATLAB-trained LSTM model into the foxBMS-1 firmware, compile it, and generate a .hex file ready to be flashed onto an STM32F4 microcontroller.

## ✅ Step-by-Step Summary

### 1. **Prepare LSTM Model in MATLAB**

* Train LSTM for SoC estimation.
* Ensure the model is configured to use double precision (default).
* Save the model as trained\_lstm.mat.

#### ▶ Code Generation in MATLAB:

cfg = coder.config('lib');  
cfg.GenCodeOnly = true;  
cfg.TargetLang = 'C';  
cfg.Hardware = coder.Hardware('STM32F4xx Based'); % Not sure, maybe commented it  
cfg.DeepLearningConfig = coder.DeepLearningConfig('TargetLibrary', 'none');  
cfg.EnableOpenMP = false; % Avoids omp.h errors  
cfg.GenerateReport = true;  
  
input\_type = coder.typeof(0, [1, inf]);  
codegen socEstimator.m -args {input\_type, input\_type, input\_type} -config cfg -report

### 2. **Integrate Generated Code into foxBMS**

#### A. Create New Folder

* Create a folder:

embedded-software/mcu-primary/src/application/sox/soc\_lstm/

* Copy all generated .c/.h files from MATLAB’s codegen/lib/socEstimator/ to this folder.
  + Exclude folders like /examples and /html

#### B. Remove rtwtypes.h

* Delete or ignore rtwtypes.h to avoid conflicts with foxBMS types.

#### C. Create matlab\_types\_extension.h

To supplement types expected by MATLAB code:

#ifndef MATLAB\_TYPES\_EXTENSION\_H\_  
#define MATLAB\_TYPES\_EXTENSION\_H\_  
  
#include <stdint.h>  
#include "matlab\_types.h"  
  
#ifndef real32\_T  
typedef float real32\_T;  
#endif  
  
#ifndef real64\_T  
typedef double real64\_T;  
#endif  
  
#ifndef int8\_T  
typedef int8\_t int8\_T;  
#endif  
  
#ifndef uint8\_T  
typedef uint8\_t uint8\_T;  
#endif  
  
#ifndef int16\_T  
typedef int16\_t int16\_T;  
#endif  
  
#ifndef uint16\_T  
typedef uint16\_t uint16\_T;  
#endif  
  
#ifndef uint32\_T  
typedef uint32\_t uint32\_T;  
#endif  
#endif  
#ifndef boolean\_T  
typedef int boolean\_T;  
#endif  
#ifndef true  
#define true 1  
#endif  
#ifndef false  
#define false 0  
#endif  
#ifndef MAX\_int32\_T  
#define MAX\_int32\_T 2147483647  
#endif  
#ifndef MIN\_int32\_T  
#define MIN\_int32\_T (-2147483647 - 1)  
#endif  
  
#endif

#### D. Update sox.c Files to Include the Above:

Include matlab types extension library:

#include "matlab\_types\_extension.h"

### 3. **Write Wrapper Function**

Create socEstimator\_wrapper.c:

#include "socEstimator.h"  
#include "socEstimator\_initialize.h"  
#include "socEstimator\_terminate.h"  
#include "socEstimator\_emxAPI.h"  
#include "socEstimator\_emxutil.h"  
  
void SOX\_GetStateOfCharge\_fromLSTM(double \*voltage, double \*current, double \*temperature, int len, double \*soc\_out) {  
 static int is\_initialized = 0;  
 if (!is\_initialized) {  
 socEstimator\_initialize();  
 is\_initialized = 1;  
 }  
  
 // Allocate arrays and call estimator  
 // float soc\_array[1024]; // adjust size as needed  
 // float step\_times[1024]; // optional, remove if unused  
  
 emxArray\_real\_T \*v\_arr = emxCreateWrapper\_real\_T(voltage, 1, len);  
 emxArray\_real\_T \*c\_arr = emxCreateWrapper\_real\_T(current, 1, len);  
 emxArray\_real\_T \*t\_arr = emxCreateWrapper\_real\_T(temperature, 1, len);  
 emxArray\_real\_T \*soc\_arr = emxCreate\_real\_T(1, len); // output  
  
 socEstimator(v\_arr, c\_arr, t\_arr, soc\_arr);  
 //socEstimator(voltage, current, temperature, soc\_array);  
  
 //\*soc\_out = soc\_array[len - 1]; // take latest estimate  
 \*soc\_out = soc\_arr->data[len - 1];  
  
 // Free arrays  
 emxDestroyArray\_real\_T(v\_arr);  
 emxDestroyArray\_real\_T(c\_arr);  
 emxDestroyArray\_real\_T(t\_arr);  
 emxDestroyArray\_real\_T(soc\_arr);  
}

Declare in socEstimator\_wrapper.h:

#ifndef SOC\_ESTIMATOR\_WRAPPER\_H  
#define SOC\_ESTIMATOR\_WRAPPER\_H  
  
void SOX\_GetStateOfCharge\_fromLSTM(double \*voltage, double \*current, double \*temperature, int len, double \*soc\_out);  
  
#endif

### 4. **Modify ``**\*\* in foxBMS\*\*

* Replace or bypass Coulomb counting logic.
* find SOC\_Calculation function and replace it with following:
* void SOC\_Calculation(void) {
* DB\_ReadBlock(&sox\_current\_tab, DATA\_BLOCK\_ID\_CURRENT\_SENSOR);
* DB\_ReadBlock(&cellminmax, DATA\_BLOCK\_ID\_MINMAX);
* float v = (float)(cellminmax.voltage\_mean) / 1000.0f;     // mV to V
* float c = (float)(sox\_current\_tab.current) / 1000.0f;     // mA to A
* float t = (float)(cellminmax.temperature\_mean);           // °C
* voltage\_buffer[buffer\_index] = v;
* current\_buffer[buffer\_index] = c;
* temperature\_buffer[buffer\_index] = t;
* buffer\_index++;
* if (buffer\_index >= LSTM\_INPUT\_LEN) {
* buffer\_index = 0;
* buffer\_full = true;
* }
* if (buffer\_full) {
* double soc\_out = 0.0;
* SOX\_GetStateOfCharge\_fromLSTM(voltage\_buffer, current\_buffer, temperature\_buffer, LSTM\_INPUT\_LEN, &soc\_out);
* // Clamp the output
* if (soc\_out > 100.0) soc\_out = 100.0;
* if (soc\_out < 0.0) soc\_out = 0.0;
* // Set SoC values
* sox.soc\_mean = soc\_out;
* sox.soc\_min = soc\_out;
* sox.soc\_max = soc\_out;
* // Store to NVM
* SOX\_SOC\_s soc\_struct = {soc\_out, soc\_out, soc\_out, 0, 0, 0, 0};
* NVM\_setSOC(&soc\_struct);
* DB\_WriteBlock(&sox, DATA\_BLOCK\_ID\_SOX);
* }
* }
* Add include:

#include "socEstimator\_wrapper.h"

* Ensure double voltage\_buffer[], current\_buffer[], and temperature\_buffer[] are used.
* Replace bool with boolean\_T, and use true/false from matlab\_types\_extension.h.
* Replace float literals 100.0f with 100.0 (for double compatibility).

### 5. Update  wscript

* Add all new .c files from soc\_lstm/ into the srcs = list of  
  embedded-software/mcu-primary/src/application/wscript
* Example:

os.path.join('sox', 'soc\_lstm', 'socEstimator.c'),  
os.path.join('sox', 'soc\_lstm', 'socEstimator\_initialize.c'),  
os.path.join('sox', 'soc\_lstm', 'socEstimator\_terminate.c'),  
os.path.join('sox', 'soc\_lstm', 'socEstimator\_data.c'),  
os.path.join('sox', 'soc\_lstm', 'socEstimator\_emxAPI.c'),  
os.path.join('sox', 'soc\_lstm', 'socEstimator\_emxutil.c'),  
os.path.join('sox', 'soc\_lstm', 'predictAndUpdateState.c'),  
os.path.join('sox', 'soc\_lstm', 'resetState.c'),  
os.path.join('sox', 'soc\_lstm', 'socEstimator\_wrapper.c'),

* Also add 'sox/soc\_lstm' to the includes += section.

***os.path.join(‘sox’, ‘soc\_lstm’),***

### 6. **Build foxBMS Firmware**

Run the following from the foxBMS *root* directory:

python tools/waf configure  
python tools/waf build\_primary

If all steps were followed correctly, you will see:

[279/279] Creating hex file ...  
'build\_primary' finished successfully

And in the ***-software-primary\*** get:

foxbms\_primary.hex

ready to flash using STM32CubeProgrammer.

## Final Result

* A fully integrated LSTM model from MATLAB inside foxBMS
* Code that compiles cleanly
* A .hex file that is ready to flash on STM32