تعریف متغیر

متغیر ها به سه دسته ی کلی Local ، Global ، Const تقسیم می گردند.

متغیر های Const به صورت UPPER\_CASE باید تعریف گردند.

static const uint32\_t SYSTEM\_VERSION = 10;

متغیر های local و global به صورت camelCase باید تعریف گردند.

int16\_t keyValue;

تحت هیچ شرایطی از انواع داده ای پیشفرض زبان C به غیر از char استفاده نگردد.

int tempValue; // -> int32\_t tempValue;

short x; // -> int16\_t x;

اگر متغیر های تعریف شده مربوط به بخش یا ماژول خاصی باشد و private یا local نباشند (Global باشند( باید اسم ماژول در ابتدای آن بیاید.

uint8\_t keypadState;

const uint8\_t KEY\_STATE\_NONE = 0;

در تعریف پوینتر کاراکتر \* باید به سمت چپ خود که نوع داده می باشد بچسبد.

uint8\_t\* pX;

char\* src;

در هر خط فقط یک متغیر تعریف گردد.

تعریف توابع

تعریف توابع با قابل زیر انجام می گردد بدین صورت که ابتدا نام ماژول یا بخش سپس نام تابع می آید. نام تابع باید camelCase باشد.

<Module>\_<function name>

void Key\_handle(void);

char Str\_compare(char\* a, char\* b);

void KeyPad\_setCallback(KeyPad\* keypad, KeyPad\_CallbackFn fn);

تعریف نوع داده ی جدید

در تعریف اونواع داده ای اعم از struct ، union و enum باید حتما به صورت PascalCase باشد و حتما به صورت typedef تعریف گردند. قالب تعریف به صورت زیر باشد.

<Module>[\_<Type Name>]

typedef struct {

// …

} KeyPad;

typedef enum {

// …

} Key\_State;

در تعریف function pointer ها حتما انتهای آن ها Fn نوشته شود.

typedef void (\*Key\_CallbackFn)(void);

ساختار ماژول (کتابخانه)

ساختار کتابخانه باید به صورتی باشد که قابلیت انتقال آن از یک میکرو به میکرویی دیگر به راحتی و باکمترین تغییرات امکان پذیر باشد. بدین منظور می تواندی از تکنیک های Driver و Callback استفاده نمایید. کد های تولید شده توسط Cube نباید با محتوایت درون کتابخانه در تعارض باشند و باید بتوان به راحتی هر کدی که تولید می شود را به کتابخانه متصل نمود.

اگر کتابخانه نیاز دارد تکه کدی یا بخشی به طور مرتب یا با ریتم منظم اجرا گردد باید درون تابعی با نام handle قرار گیرد و در بخش کامنت آن باید فاصله ی زمانی بین هر بار اجرا شدن این تابع مشخص و به کاربر اعلام گردد.

کامنت گذاری

کامنت گذاری برای تمام توابع در فایل C باید وجود داشته باشد به صورت English یا Finglish و از ساختار Doxygen باید پیروی کنند.

برای فیلد های درون struct، enum و union نیز باید کار آن فیلد نوشته شود.

برای تمام define ها نیز باید کامنت گذاری انجام گردد.

تعریف define

تعریف define ها به سه دسته ی کلی تقسیم می گردند.

* به عنوان Config یا Label
* به عنوان ماکرو به صورت global
* به عنوان ماکرو به صورت private

نحوه ی نام گذاری define های Config باید به صورت UPPER\_CASE باشد.

#define KEY\_CONFIG

نحوه ی نام گذاری ماکرو های global باید به صورت camelCase و دقیقا مطابق نگارش توابع باشند.

#define IStream\_available(STREAM)

نحوه ی نام گذاری ماکرو های private بسته به نوع کار برد می تواند شبیه توابع باشد اما بهتر است به صورت camelCase و به همراه دو کاراکتر under line (\_\_) نوشته شود.

#define \_\_updateKey(K)

تنظیمات پروژه

تنظیمات پروژه حتما باید بر روی Optimization 0 در بخش C/C++ یا نهایت 1 باشد (در صورتی که بر روی 0 به درستی کار کرده باشد)

تیک GNU Extensions باید فعال باشد.