PUBLIC | Principy registrové mapy

Princip

Zakladni principy registrove mapy jsou:

- Naskladani promennych do linearniho pametoveho prostoru tak, ze kazda promenna je umistena na nasledujici volnou adresu za predchozi.
- Prommene jsou sdruzovany do bloku, jednotlive bloky nemusi byt hned za sebou. To umoznuje, ze rozsireni jednoho bloku neovlivni ostatni bloky.
- Kazda promenna ma prirazeny identifikator, z nehoz je jasne urcitelna adresa v pameti (tedy o jaky blok se jedna a jaky je offset v ramci bloku).
- Identifikator obsahuje i delku dat
- Identifikator umoznuje genericke nacteni/zapsani promenne -> jednoduche protokolarni parsery

Struktura Identifikatoru

Identifikator je dlouhy 32 bitu a ma nasledujici strukturu:

Bits position	Name	Description
31 - 24	Block	Identification of functional block
23 - 12	Address	Byte address of the beginning of variable within block
11 - 8	Data Type	ID of type (int, string, float, byte[])
7 - 5	Access	Read/write/non-volatile access right
4	Extended ID	Always 1 – 32-bit extended ID
3 - 0	Length	Exponent of variable length (0 -> 1B, 1 -> 2B, 2 -> 4B, 3 -> 8B, 4 -> 16B, 5 -> 32B, 6 -> 64B, etc. 14 and 15 reserved for variable length)

Block

Identifikátor funkčního bloku, do kterého patří proměnná. Umožňuje třízení dle bloků a zároveň umístění v paměti zařízení na vedlejších adresách. Je možno rozšiřovat do maximálního počtu 256 bloků.

Address

Definuje počáteční adresu proměnné v rámci každého funkčního bloku. To v zařízení umožní jednoduchou indexaci proměnných v paměťovém prostoru, ale musí být zaručené, že adresy dvou sousedních proměnných se liší přesně o délku té první. Toto by ale mělo být zajištěno externím nástrojem generujícím paměťovou mapu.

Address musí v rámci bloku začínat na hodnotě 0 a každá následující hodnota musí být přesně o délku předchozí proměnné větší než předchozí. Jedná se adresu v packed struktuře.
Pozor musí být dán na zarovnání proměnných v paměti vzhledem k použité architektuře. Např. M0 neumí nezarovnané přístupy pro 32 bitové inty, M4 umí inty ale neumí floaty.

Data Type

Tato část definuje datový typ proměnné. Pro samotný protokol přenosu dat to není vůbec důležité, ale pro unifikovaný přístup k datům v rámci Frontendu i mikro kontroléru se tato informace hodí. Platí zde určitá logická omezení na délku proměnných, například Float musí mít délku 4 byty, integer nemůže být delší než 8 bytů, pouze Binary a String mohou být delší než 8 bytů. Enumerace je vždycky nejkratší možné délky, tedy pokud nemá přiřazené hodnoty, které mají velkou hodnotu, je délka enumerace 1 B.

Zkratka	Hodnota	Popis
BIN	0	Binarni promenna s HEX reprezentaci
INT	1	Cele cislo - dekadicka reprezentace
FLOAT	2	4B desetinne cislo
STRING	3	Textovy retezec zakonceny nulou
IP	4	4B IP adresa
ENUM	5	Vyctovy typ, jednotlive hodnoty musi byt uvedeny

Access

Toto pole definuje, jaké operace jsou s touto proměnnou dovolené. Proměnná může mít povolenou operaci zápisu a může se automaticky ukládat do Flash paměti, její hodnota je obnovena po výpadku napájení.

Zkratka	Hodnota	Popis
RO	0	Read only - pouze operace cteni je mozna
ROF	1	Read-only-flash - pouze cteni je mozne, hodnota je nevolatilni
RW	2	Read-write - lze cist i zaposivat, hodnota je volatilni
RWF	3	Read-write-flash - cteni i zapis, hodntoa ulozena v nevolatiln flash
RWIF	3 (! ne 4)	Read-write-internal-flah - cteni i zapis, hodnota ulozena v interni flash

Length

Poslední část identifikátoru proměnné definuje délku proměnné. Pro jednoduchost jsou zatím všechny proměnné konstantní délky, přestože je zamýšlená rozšiřitelnost na proměnnou délku. To však bude realizováno pouze v případě potřeby. Délka je kódována jako exponent mocniny dvou, čili 0 znamená 1 byte, 1 značí 2 byty, 2 = 4 byty, 3 = 8 bytů atd. Hodnoty 14 a 15 jsou rezervovány pro variabilní délky, jejich použití zatím není očekávané.

Typ Enum

Vyctovy typ je dost specificky typ promenne, jejiz definice zabira vice nez jeden radek. V prvnim radku se samotna definice promenne tak, jak jsme zvykli. Musi byt vyplneny sloupec typu, access code, adresa, velikost, vysledny nazev. Na *x* nasledujicich radcich jsou pote uvedeny jednotlivy vyctove polozky a jejich hodnoty. Zde musi platit, ze neni uveden data type, Name obsahuje nazev vyctu v kodu, Label obsahuje popisek pro JSONy a Factory value obsahuje hodnotu. Pokud neni hodnota uvedena, priradi se inkrementalne hodnoty od 0 do *x-1*. Delka promenne musi souhlasit s vyslednou delku enumu v C, coz se muze lisit prekladac od prekladace a v zavislosti na nejvetsi uvedene hodnote. ARM ma typicky 1 B.

Samotna promenna typu enum muze mit defaultni hodnotu uvedenou ciselnou hodnotou.

Priklad

MB_BAUD_RATE	COM	ENUM	RWF	1	1	0x06001570	COM_MB_BAUD_RATE	Modbus baud rate	4
	COM			2		0	MB_BAUD_9600	9600	
	COM			2		0	MB_BAUD_19200	19200	
	COM			2		0	MB_BAUD_38400	38400	
	COM			2		0	MB_BAUD_57600	57600	
	COM			2		0	MB_BAUD_115200	115200	

Typ Bin

Tento typ je ekvivalent typu Integer (INT) pouze s tim rozdilem, ze muze byt delsi nez 8 B a zaroven muze mit definovany vycet vyznamu jednotlivych bitu. To se pote hodi pro popis promennych obsahujici bitove masky nebo soubor bitu, kdy kazdy ma jiny vyznam - STATUS registr, CONTROL registr atd. Tato definice dalsich bitu neni povinna, viz:

CPU_TEMPER	SYS	FLOAT	RO	12	4	0x0000C212	SYS_CPU_TEMPER	CPU temperature	25,1
PASSWORD_HASH	SYS	BIN	ROF	16	8	0x00010033	SYS_PASSWORD_HASH	Password hash	0
TEST	SYS	INT	RW	24	4	0x00018152	SYS_TEST	Testing register	3

kde BIN promenna nema definovany vyznam bitu (jedna se o 8 B hash)

Ale pokud chceme, muzeme si vyznam bitu definovat stejnym zpusobem, jako jsme si definovali vyctovy typ. Tedy Name obsahuje identifikator bitove pozice pro C kod, Label obsahuje popisek daneho bitu a Factory value obsahuje bitovou pozici (indexovano od nuly). V pripade prazdne kolonky factory value se bitove pozice inkrementuji automaticky.

STATUS	REGU	BIN	RW	16	4	0x04010052	REGU_STATUS	Status register	16
	REGU			20		0	STATUS_ERROR	General error	0
	REGU			20		0	STATUS_TIMEOUT	Modbus timeout	4
	REGU			20		0	STATUS_BUSY	Device is busy	5
	REGU			20		0	STATUS_WARNING	General warning	6
DUMMY	REGU	INT	RW	20	4	0x04014152	REGU_DUMMY	Dummy variable	1

Toto ve vysledku vygeneruje do reg_map.h soubor definu:

```
#define STATUS_ERROR                (1 << (0))
#define STATUS_TIMEOUT              (1 << (4))
#define STATUS_BUSY                 (1 << (5))
#define STATUS_WARNING              (1 << (6))
```

A do JSONU soubor popisku

```
    "EnumStr": ["General error", "Modbus timeout", "Device is busy", "General warning" ],
    "EnumValue": [ 0, 4, 5, 6],
```