Soft & Control Technology s.r.o.

Batériové bezdrôtové telemetrické systémy radu BaWiT

SPL ASM

Programovací jazyk
Používateľská príručka

Verzia dokumentu 1.1
Revízia 1.10
Dátum 18/6 2014

Obsah

1 Pojmy

SPL ASM - SCT PLCprogramming Language ASM.
UDALOSŤ - Udalosť jazyka ktorá má pridelenú obsluhu.
Interpretovať - Vykonávať inštrukcie zakódované pomocou Bytecode.
Kompilovať - Vytvoriť ByteCode z zdrojového súboru.
ByteCode - Zakódované inštrukcie, ktoré sú priamo interpretovateľné v zariadení s podporou SPL.

2 Jazyk SPL ASM

Jazyk SPL ASM je nízkoúrovňový jazyk, ktorý umožňuje písať obsluhu udalostí pre zariadenia typu BaWiT. Udalosti sa generujú pri spracovaní dátových bodov, obsluhe zariadení, činnosti systémových prostriedkov alebo sa generujú priamo spracovaním inštrukcií jazyka SPLASM. Pomocou vývojového prostredia jazyka sa kompiláciou vytvorí BYTECODE, ktorý je súčasťou konfigurácie zariadenia. Teakto vygenerovaný kód je priamo vykonávateľný interpreterom v BaWiT-e.

3 Popis udalosti

Udalosti sa generujú pri spracovaní dátových bodov:

- Dosiahnutie/Prekročenie užívateľských hraníc dátového bodu (analógové aj digitálne)
- Nová hodnota (načítaná) môže byť aj rovnaká ako predchádzajúca
- Zmenená hodnota (o citlivosť od vygenerovania udalosti)
- Stabilná hodnota (hodnota sa nemení v stanovenom čase podľa konfigurácie)

Systémové udalosti

- Reštart zariadenia
- Prechod z režimu SLEEP do režimu RUN
- Zapnutie modemu, vypadnutie modemu, ...

Udalosti jazyka

- Vypršanie časovača jazyka
- Dosiahnutie času alarmu (budíka)
- Priame zadanie udalosti

4 Vykonávanie jazyka

Ak sa v zariadení vygeneruje udalosť, zaradí sa do fronty pre spracovanie. PowerManager v rámci optimalizovania spotreby určuje začiatok interpretovania udalosti v čo najkratšom čase ak sú dostupné všetky prostriedky pre spracovanie. Udalosť sa spracováva paralelne s vykonávaním všetkých ostatných úloh podľa konfigurácie.

5 Obmedzenia

Počet registrov je obmedzený na 8. Počet súčtov môže byť max 8. Počet vnorení if-if-else-else môže byť max. 8. Počet časovačov je 8. Počet alarmov je 8.

5.1 Použité dátové typy

UINT – 32 bitový bezznamienkový celočíselný typ
INT – 32 bitový znamienkový celočíselný typ
FLOAT – 32 bitový typ podľa IEEE754.

6 Preprocesorové príkazy

Aj keď sa nasledovné príkazy označujú ako preprocesorové, nejedná sa o klasický preprocesor (ako napr. v jazyku C). Spracovanie zdrojových súborov nie je rozdelené na preprocesor a následný kompilátor; je iba kompilátor, ktorý postupne spracováva jednotlivé príkazy a inštrukcie. Ako oddeľovač jednotlivých príkazov slúži znak konca riadku. Jazyk nerozlišuje veľké a malé písmená

6.1 #CONST

Definuje konštantu, ktorú je možné použiť vo výrazoch. Hodnota konštanty musí byť vypočítateľná už počas kompilácie.

```
#CONST constant = constexpression {, constant = constexpression }
```

Konštanta môže byť definovaná iba mimo tela akcie. Už raz definovaná konštanta môže byť znovu predefinovaná a v ďalšom spracovaní zdrojového súboru sa bude pracovať s jej novou hodnotu (príp. typom). Konštanty sú viditeľné aj mimo aktuálneho zdrojového súboru (viď príkaz `#INCLUDE`).

Typ konštanty je automaticky určený kompilátorom zo zadaného konštantného výrazu (uint, int, float, string).

Príklad:

```
#CONST i = 1, j = i + 1, k = -1
#CONST tlak = "FCGPA_"
#CONST tlak1 = tlak + "1", tlak2 = tlak + "2"
#CONST x = 3.5234, y = x + i + 0.5
```

V príklade vyššie budú mať konštanty nasledovný typ:

Konštanta	Typ
i, j	uint
k	int
x, y	float
tlak, tlak1, tlak2	string

6.2 #ACTION … #END

Definuje užívateľskú akciu.

```
#ACTION ActionName EventId EventIndex
instructions
#END
```

`ActionName` – názov akcie, ktorý bude vyplnený v *09 – UserActions* / *UserName*. `EventId` – typ eventu, ktorý bude vyplnený v *09 – UserActions* / *UserEvent*. EventIndex – index eventu, ktorý bude vyplnený v *09 – UserActions* / *UserEventIndex*. Ako `EventId` a `EventIndex` je možné použiť kladnú celočíselnú hodnotu alebo identifikátor celočíselnej konštanty.

Akcie sú viditeľné aj mimo aktuálneho zdrojového súboru (viď príkaz `#INCLUDE`).

6.3 #DEBUG

Zapína/vypína nastavenie debug a break bitu v inštrukciách.

```
#DEBUG [ON | OFF] [BREAK]
```

Príkaz môže byť použitý vnútri alebo mimo tela akcie. Použitím parametra `BREAK` bude prvá nasledujúca inštrukcia vygenerovaná s nastaveným BREAK bitom. Použitím parametrov `ON` / `OFF` sa budú všetky nasledujúce inštrukcie generovať s príslušne nastaveným DEBUG bitom. Príkaz má oblasť pôsobnosti len v aktuálnom zdrojovom súbore.

V príklade nižšie budú inštrukcie MOV, ADD, DIV, MUL mať nastavený DEBUG bit na 1. Inštrukcia DIV bude mať zároveň nastavený BREAK bit na 1. Inštrukcie NEW a SUB nebudú mať nastavený žiaden zo spomínaných bitov. Použitie #DEBUG bez parametrov je kompilátorom ignorované.

Príklad:

```
NEW
#DEBUG ON
MOV CNT[0], 5
ADD CNT[0], CNT[0], CNT[1]
#DEBUG BREAK
DIV CNT[0], CNT[0], CNT[2]
MUL CNT[0], CNT[0], 2
#DEBUG OFF
SUB CNT[0], CNT[0], CNT[3]
```

6.4 #RESULT

Určuje prednastavený typ výsledku inštrukcií MOV, ADD, SUB, MUL, DIV, MOD, AND, IOR a XOR, pokiaľ typ výsledku nie je uvedený priamo pri inštrukcii.

```
#RESULT UINT | INT | FLOAT
```

Príkaz je možné použiť vnútri aj mimo tela akcie. V oboch prípadoch funguje ako globálne nastavenie. Platnosť príkazu je od miesta jeho výskytu až po jeho ďalší výskyt, príp. po koniec súboru. Príkaz má oblasť pôsobnosti len v aktuálnom zdrojovom súbore.

6.5 #PARAM

Slúži na nastavenie dodatočných konfiguračných parametrov akcie. Pokiaľ je príkaz použitý mimo tela akcie, má vplyv na všetky ďalej definované akcie. Ak je príkaz použitý vnútri tela akcie, má vplyv na akciu, v ktorej je použitý a zároveň na všetky ďalšie akcie. Príkaz má oblasť pôsobnosti len v aktuálnom zdrojovom súbore.

```
#PARAM name constexpression
```

V súčasnosti sú podporované nasledovné parametre (name): LogDestination, LogNumber, LogSize. Konštantný výraz constexpression musí byť celočíselný.

6.6 #INCLUDE

Slúži na vsunutie a následné spracovanie obsahu iného zdrojového súboru do aktuálne spracovávaného zdrojového súboru.

```
#INCLUDE <filename>
```

Pri spracovaní vsúvaného súboru budú viditeľné všetky dovtedy definované konštanty a akcie. Po skončení spracovania vsúvaného súboru budú v nadradenom zdrojovom súbore viditeľné všetky novo definované konštanty (príp. zmenené hodnoty predtým definovaných konštánt) a akcie. Medzi nadradeným a vsúvaným súborom sa neprenášajú stavy určené príkazmi #DEBUG, #PARAM a #RESULT.

Vsúvaný súbor je hľadaný v systémových adresároch definovaných kompilátorom.

Príklad:

Súbor A

```
#CONST X = 1 << MOC
#DEBUG ON
```

Súbor B

```
#DEBUG OFF
#CONST MOC = 2
#INCLUDE <A>
```

```
#CONST Y = X - 1
```

V súbore *B* je možné od vsunutia súboru *A* používať konštantu `X` definovanú v súbore *A*. Príkaz `#DEBUG ON` nemá na súbor *B* žiadny vplyv, podobne príkaz `#DEBUG OFF` nemá žiaden vplyv na súbor *A* (pri spracovaní súboru *A* sa použije východzie nastavenie kompilátora). V súbore *A* sa používa konštanta `MOC`, ktorá nie je v súbore *A* definovaná, je však definovaná skôr, ako dôjde ku spracovaniu súboru *A*.

6.7 #STRING, #SHORTSTRING, #LONGSTRING

Slúži na definíciu textov, ktoré sa generujú pri kompilácii zdrojového kódu.

```
#STRING id, text
#SHORTSTRING id, text
#LONGSTRING id, text
```

Kompilátor pri každej tejto direktíve vygeneruje text so zadaným id, ktorý je potom uložený do výsledného výstupu kompilátora. Ako `id` je možné zadať ľubovoľný konštantný výraz, ktorého výsledkom je nezáporné celé číslo. Ako `text` je možné zadať ľubovoľný konštantný výraz, ktorého výsledkom je reťazec.
Direktíva `#SHORTSTRING` určuje, že výsledný text môže byť dlhý maximálne 8 znakov, podobne pre `#LONGSTRING`, kde môže byť výsledná dĺžka maximálne 64 znakov.
Direktíva `#STRING` určuje, že veľkosť výsledného textu bude daná nastavením kompilátora (vždy krátky text, vždy dlhý text alebo automaticky).

6.8 #VALUE

Slúži na definíciu hodnôt, ktoré sa generujú pri kompilácii zdrojového kódu a sú uložené v konfigurácii zariadenia.

```
#VALUE id, ident = (UINT | INT | FLOAT) value
```

Kompilátor pri každej tejto direktíve vygeneruje hodnotu so zadaným id, ktoré je potom uložená do výsledného výstupu kompilátora. Ako `id` je možné zadať ľubovoľný konštantný výraz, ktorého výsledkom je nezáporné celé číslo. Ako `value` je možné taktiež zadať ľubovoľný konštantný výraz. Pre každú hodnotu je potrebné explicitne určiť typ pod akým bude uložená v konfigurácii.

7 Komentáre

Na písanie komentárov slúži znak bodkočiarka (`;`) alebo dve lomítka (`//`). Každý znak za bodkočiarkou alebo dvoma lomítkami až do konca riadku je kompilátorom ignorovaný.

8 Konštantné výrazy

Konštantné výrazy sú výrazy, ktoré je možné vypočítať už v dobe kompilácie zdrojového súboru. Pozostávajú z definovaných konštánt (`#CONST`) alebo priamo zadaných číselných hodnôt.
V konštantných výrazoch je možné používať nasledujúce operátory.

`+ - * /`	Sčítanie, odčítanie, násobenie a delenie.
`\ %`	Celočíselné delenie a zvyšok po celočíselnom delení.
`& | ^`	Bitový súčin, bitový súčet a exkluzívny bitový súčet.
`~`	Bitová negácia.
`<< >>`	Bitový posun vľavo a vpravo.

Operátory `+ -  |` majú menšiu prioritu ako ostatné operátory. Operátory s rovnakou prioritou sú spracovávané v poradí v akom sa vyskytli vo výraze. Poradie spracovávania operátorov je možné ovplyvniť použitím zátvoriek `( )`.
Ďalej je možné v konštantných výrazoch používať nasledovné funkcie:

DT (datetime)

Prevedie zadaný dátum a čas `datetime` na jeho celočíselnú reprezentáciu (počet sekúnd od 1.1.2000 00:00:00). Parameter `datetime` musí byť typu string a musí byť zadaný v tvare `rrrr-mm-dd hh:mm:ss`). V reťazci je možné vynechať dátumovú alebo časovú časť. V prípade vynechania dátumovej časti sa použije dátum 1.1.2000. Pri vynechaní časovej časti sa použije čas 00:00:00.

9 Štruktúra zdrojového súboru

Zdrojový súbor pozostáva z definície konštánt a jednotlivých akcií. Telo akcie môže obsahovať niekoľko inštrukcií.

9.1 Inštrukcie jazyka

9.1.1 END

Nepodmienené ukončenie vykonávania kódu akcie.

9.1.2 NEWIF

Začiatok novej podmienky.

9.1.3 OR

Ďalší súčtový člen podmienky.

9.1.4 NOP

Žiadna operácia.

9.1.5 ENDC

Podmienené ukončenie vykonávania kódu akcie.

9.1.6 EQ [INT | UINT | FLOAT] Src1, Src2

Porovnanie operandov (Src1 = Src2). Ako Src môže byť použitý jeden z registrov alebo konštantný výraz.

9.1.7 NE [INT | UINT | FLOAT] Src1, Src2

Porovnanie operandov (Src1 <> Src2). Ako Src môže byť použitý jeden z registrov alebo konštantný výraz.

9.1.8 LT [INT | UINT | FLOAT] Src1, Src2

Porovnanie operandov (Src1 < Src2). Ako Src môže byť použitý jeden z registrov alebo konštantný výraz.

9.1.9 LE [INT | UINT | FLOAT] Src1, Src2

Porovnanie operandov (Src1 <= Src2). Ako Src môže byť použitý jeden z registrov alebo konštantný výraz.

9.1.10 GT [INT | UINT | FLOAT] Src1, Src2

Porovnanie operandov (Src1 > Src2). Ako Src môže byť použitý jeden z registrov alebo konštantný výraz.

9.1.11 GE [INT | UINT | FLOAT] Src1, Src2

Porovnanie operandov (Src1 >= Src2). Ako Src môže byť použitý jeden z registrov alebo konštantný výraz.

9.1.12 ELSE

Začiatok inštrukcí ktoré sa vykonajú pri nesplnení podmienky.

9.1.13 ENDIF

Ukončenie inštrukcií podmienky.

9.1.14 TRUE

Vždy pravda.

9.1.15 EE Ident Value

Vyvolanie udalosti (akcie obsluhujúcej udalosť) s názvom Ident a doplňujúcim parametrom Value. Akcia obsluhujúca udalosť sa nevykoná okamžite, ale je len zaradená do fronty vyvolaných udalostí, kde čaká na spracovanie.

9.1.16 MOV [INT | UINT | FLOAT] Dest, Src1

Nastaví hodnotu operandu Dest na hodnotu operandu Src1. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.17 ADD [INT | UINT | FLOAT] Dest, Src1, Src2

Sčíta hodnoty operandov Src1 a Src2 a výsledok uloží do Dest. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.18 SUB [INT | UINT | FLOAT] Dest, Src1, Src2

Urobí rozdiel hodnôt operandov Src1 a Src2 a výsledok uloží do Dest. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.19 MUL [INT | UINT | FLOAT] Dest, Src1, Src2

Vynásobí hodnoty operandov Src1 a Src2 a výsledok uloží do Dest. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.20 DIV [INT | UINT | FLOAT] Dest, Src1, Src2

Vydelí hodnotu Src1 hodnotou Src2 a výsledok uloží do Dest. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.21 MOD [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží zvyšok po delení hodnoty Src1 hodnotou Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.22 AND [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží výsledok bitového súčinu Src1 a Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.23 IOR [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží výsledok bitového súčtu Src1 a Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj

požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.24 XOR [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží výsledok exkluzívneho bitového súčtu Src1 a Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.25 DIF [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží absolútnu hodnotu rozdielu Src1 a Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.26 MEQ [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží hodnotu výrazu Src1==Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.27 MNE [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží hodnotu výrazu Src1!=Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.28 MGT [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží hodnotu výrazu Src1>Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.29 MGE [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží hodnotu výrazu Src1>=Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.30 MLT [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží hodnotu výrazu Src1<Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.31 MLE [INT | UINT | FLOAT] Dest, Src1, Src2

Do Dest uloží hodnotu výrazu Src1<=Src2. Ako Dest môže byť použitý jeden z registrov, ako Src môže byť použitý jeden z registrov alebo konštantný výraz. . V prípade, že je uvedený aj požadovaný typ výsledku, bude vysledná hodnota uložená so špecifikovaným typom. V opačnom prípade sa použije nastavenie podľa posledného príkazu `#RESULT`.

9.1.32 JMP label

Urobí skok na inštrukciu označenú label. Skoky je možné robiť len v rámci jednej akcie. V dvoch rôznych akciách je možné použiť tie isté označenia pre miesto skoku. Inštrukcia `JMP` sa preloží ako

```
MOV UINT STS[0], absolute_offset
```

Hodnotu `absolute_offset` vypočíta kompilátor. V súčasnosti neexistuje inštrukcia pre skok o relatívny offset. Je treba použiť `ADD UINT STS[0], STS[0], relative_offset`.
V príklade nižšie urobí inštrukcia `JMP` skok na inštrukciu `ADD`.

Príklad pre JMP:

```
MOV CNT[0], DP[0]       // Nastavi Counter0 na hodn. dat. bodu s adresou 0
miesto:                 // Návestie
ADD CNT[0], CNT[0], 5   // Pripočíta do Counter0 hodnotu 5
MUL CNT[0], CNT[0], 2   // Vynásobí Counter0 konštantou 2
NEW                     // Zaciatok novej podmienky
LT CNT[0], 100          // Podmienka ci je hodnota Counter0 mensia ako 100
JMP miesto              // Skok na navestie miesto
NEW
MOV DP[0], CNT[0]       // Nastavi hodn. dat. bodu s adresou 0 hodn. Counter0
```

9.2 Registre

NO	Bez operandu. Implicitne sa vyhodnotí ako UINT 0.
W[*const***]**	Dočasné pomocné registre akcie. *const* – celočíselný konštantný výraz
DP[*addr***]** **DP[***shortname***]**	Vlastné hodnoty dátových bodov. *addr* – 2-bytová adresa dátového bodu (celočíselný konštantný výraz) *shortname* – skrátené označenie dátového bodu (reťazec alebo identifikátor reťazcovej konštanty)
DP[*addr***].***constident*** **DP[***shortname***].***constident***	Konfiguračné hodnoty dátových bodov. *addr* – 2-bytová adresa dátového bodu (celočíselný konštantný výraz) *shortname* – skrátené označenie dátového bodu (reťazec alebo reťazcová konštanta) *constident* – identifikačné číslo parametra konfiguračnej položky dátového bodu (kladné celé číslo alebo identifikátor celočíselnej konštanty) • 0 – vlastná hodnota • 1 – SCT time • 2 – Quality Descriptor • Iné – parametre dátového bodu podľa číselníka
CFG[*area,item_index,parameter***]**	Hodnota parametra *parameter* konfiguračnej tabuľky v oblasti *area* v položke s poradovým číslom *item_index*. *area* – identifikačné číslo oblasti konfigurácie (celočíselný konštantný výraz) • 1 – Objects • 2 – Slave devices • … *item_index* – poradové číslo položky v konfigurácii

	(celočíselný konštantný výraz) *parameter* – identifikačné číslo parametra v konfigurácii (celé číslo alebo identifikátor celočíselnej konštanty)
STS[*table,item,parameter***]**	Hodnota parametra *parameter* stavovej tabuľky s id číslom *table* v položke s id číslom *item*. *table* – identifikačné číslo stavovej tabuľky (celočíselný konštantný výraz) *item* – identifikačné číslo položky v stavovej tabuľke (celočíselný konštantný výraz) *parameter* – identifikačné číslo parametra v konfigurácii (celé číslo alebo identifikátor celočíselnej konštanty)
CNT[*const***]**	Počítadlo.
CTC[*const***]**	Časovač.
ALR[*const***]**	Alarm.

Ako indexy pre prístup na konkrétne hodnoty registrov musia byť použité iba konštantné výrazy (tzn. musia byť vypočítateľné v dobe kompilácie).

10 Gramatika

Gramatika je typu LL(1)[1] a je napísaná v mierne upravenej EBNF[2]. Štartovací neterminál gramatiky je **SPLAsm**.

```
EOLN      = '\n' | '\r' .
Sign      = "+" | "-" .
Digit     = "0".."9" .
HexDigit  = "0".."9" | "A".."F" | "a".."f" .
Letter    = "_" | "A".."Z" | "a".."z" .
Exponent  = "E" | "e" .
noQuote   = ANY - '"' - EOLN .
noAngleBr = ANY - '<' - '>' - EOLN .
ident     = Letter { Letter | Digit } .
integer   = (Digit { Digit }) | ("0x" HexDigit {HexDigit}) .
real      = [ Digit { Digit } ] "." Digit { Digit } [ Exponent [ Sign ] Digit { Digit } ] .
string    = '"' { noQuote | '\"' } '"' .
angleStr  = '<' { noAngleBr } '>' .
SPLAsm  = { Block } EOF .
Comment = (";" | "//") {ANY} .
Block = [ ConstStmt | ValueStmt | StringStmt | ActionStmt | DebugStmt | ResultStmt | ParamStmt |
        IncludeStmt ] [Comment] EOLN .
ConstStmt = "#CONST" ConstDecl { "," ConstDecl } .
ConstDecl = ident "=" ConstExpr .
ConstExpr = ConstTerm { ("+" | "-" | "|") ConstTerm } .
ConstTerm = ConstFactor { ("*" | "/" | "%" | "&" | "^" | "<<" | ">>" ) ConstFactor } .
ConstFactor = ident | integer | real | string | "DT" "(" ConstExpr ")" |
        ( "(" ConstExpr ")" ) | ("~" ConstFactor) |  ( ("+" | "-" ) ConstFactor ) .
ValueStmt = "#VALUE" ConstExpr , ident = ResultType ConstExpr .
StringStmt = ("#STRING" | "#SHORTSTRING" | "#LONGSTRING") ConstExpr "," ConstExpr .
ResultStmt = "#RESULT" ResultType .
ResultType = "UINT" | "INT" | "FLOAT" .
DebugStmt = "#DEBUG" ["ON" | "OFF"] ["BREAK"] .
ParamStmt = "#PARAM" ident ConstExpr .
IncludeStmt = "#INCLUDE" angleStr .
ActionStmt = "#ACTION" ActionHeader { Command } "#END" .
ActionHeader = ident (ident | integer) (ident | integer) [Comment] EOLN .
Command = [ (ident ":") | Instruction | DebugStmt | ResultStmt | ParamStmt ] [Comment] EOLN .
Instruction = "END" | "NEWIF" | "OR" | "ELSE" | "TRUE"
    | ( ("EQ" | "NE" | "LT" | "LE" | "GT" | "GE") [ResultType] Operand "," Operand)
    | "NOP" | "ENDC"
    | ( "EE" ident Operand )
```

[1] Typ analýzy vstupu.

[2] EBNF – Rozšírená Backus-Naur forma zápisu.

```
    | ( "MOV" [ResultType] DestOperand "," Operand )
    | ( ("ADD" | "SUB" | "MUL" | "DIV" | "MOD" | "AND" | "IOR" | "XOR"
         | "DIF" | "MEQ" | "MNE" | "MGT" | "MGE" | "MLT" | "MLE" )
        [ResultType] DestOperand "," Operand "," Operand )
    | ( "JMP" ident ) .
Operand = DestOperand | ConstExpr .
DestOperand = RegisterDP | RegisterCFG | RegisterSTS | RegisterOther | RegisterNO .
RegisterArray = "[" ConstExpr "," ConstExpr "," ConstExpr "]" .
RegisterDP = "DP" "[" ConstExpr "]" ["." (ident | integer)] .
RegisterCFG = "CFG" RegisterArray .
RegisterSTS = "STS" RegisterArray .
RegisterOther = ("W" | "CNT" | "CTC" | "ALR") "[" ConstExpr "]" .
RegisterNO = "NO" .
```

ľubovoľný znak	`ANY`
koniec vstupu	`EOF`
definícia neterminálu	`=`
alebo	`|`
voliteľné	`[...]`
opakovanie >= 0	`{...}`
zoskupenie	`(...)`
terminál	`"..." '...'`
ukončenie pravidla	`.`
interval / množina	`..`
vylúčenie	`-`

Tab. 1 Význam znakov používaných v EBNF

11 Ladenie

Na každej inštrukcii je možné zapnúť či sa daná inštrukcia podieľa na podrobnom debug výstupe. Pri spustení užívateľskej udalosti je vygenrovaný súbor v súborovom systéme BaWiT-u v adresári 0:/SPL s názvom splABCD.log, kde ABCD obsahuje hexadecimálne číslo používateľom definovanej udalosti.
Záznam potom vyzerá takto:

```
SPL Run Action ID=1 DT=2009-09-04 10:07:27 BRK=0 STS=0 Name=ZmenaPoctuBatt
    Event=48, Index=44, Param=1
SPL INS: EXE_MOV DEBUG CndTst=1 ExeTst=1 Run=1
    DST: W[3].F=1.000000
    SRC1:DP[A=1,O=0,T=0,B=0,I=0].F=1.000000 Text=batts VelID=5 <Batts>
    SRC2:NO.F=0.000000
    W:   [0]=unk [1]=unk [2]=unk [3]=1.000000 [4]=unk [5]=unk [6]=unk [7]=unk
    CNT: [0]=unk [1]=unk [2]=unk [3]=unk [4]=unk [5]=unk [6]=unk [7]=unk
    CTC: [0]=0 [1]=0 [2]=0 [3]=0 [4]=0 [5]=0 [6]=0 [7]=0
    ALR: [0]=0 U [1]=0 U [2]=0 U [3]=0 U [4]=0 U [5]=0 U [6]=0 U [7]=0 U
      T: [0]=x01.0
SPL INS: EXE_DIV DEBUG CndTst=1 ExeTst=1 Run=1
    DST: W[3].F=0.010000
    SRC1:W[3].F=1.000000
    SRC2:CONS.F=100.000000
    W:   [0]=unk [1]=unk [2]=unk [3]=0.010000 [4]=unk [5]=unk [6]=unk [7]=unk
    CNT: [0]=unk [1]=unk [2]=unk [3]=unk [4]=unk [5]=unk [6]=unk [7]=unk
    CTC: [0]=0 [1]=0 [2]=0 [3]=0 [4]=0 [5]=0 [6]=0 [7]=0
    ALR: [0]=0 U [1]=0 U [2]=0 U [3]=0 U [4]=0 U [5]=0 U [6]=0 U [7]=0 U
      T: [0]=x01.0
SPL INS: EXE_MOV DEBUG CndTst=1 ExeTst=1 Run=1
    DST: DP[A=1,O=0,T=1,B=0,I=0].F=0.010000 Text=ndb VelID=12 <NDB>
    SRC1:W[3].F=0.010000
    SRC2:NO.F=0.000000
    W:   [0]=unk [1]=unk [2]=unk [3]=0.010000 [4]=unk [5]=unk [6]=unk [7]=unk
    CNT: [0]=unk [1]=unk [2]=unk [3]=unk [4]=unk [5]=unk [6]=unk [7]=unk
    CTC: [0]=0 [1]=0 [2]=0 [3]=0 [4]=0 [5]=0 [6]=0 [7]=0
    ALR: [0]=0 U [1]=0 U [2]=0 U [3]=0 U [4]=0 U [5]=0 U [6]=0 U [7]=0 U
      T: [0]=x01.0
SPL action Time=1709msec
```

Ak nie je povolené logovanie, je uložený iba záznam o spustení užívateľskej udalosti:

```
SPL Run Action ID=0 DT=2009-09-02 14:39:58 BRK=0 STS=0 Name=Reset
    Event=16, Index=1, Param=0
SPL action Time=0msec
```

12 Príklady

12.1 Ovládanie chladenia

Zadanie
Strážiť používateľsky nataviteľnú hranicu teploty. Pri prekročení hranice zapnúť ventilátor chladenia, kontrolovať činnosť ventilátora a indikovať spustený ventilátor a indikovať chybu chladenia po každom spustení ventilátora (najskôr 5 sekúnd od zapnutia).
Popis technického riešenia
Meranie teploty pomocou PT1000 na analógovom vstupe 0. Chod ventilátora chladenia je indikovaný na digitálnom vstupe 0. Ventilátor je spúšťaný pomocou digitálneho výstupu 0. Indikácia chyby je na digitálnom výstupe 1.
Popis konfigurácie
V konfigurácii je potrebné mať správne nakonfigurované 4 dátové body podľa popisu technického riešenia. Pre správnu činnosť podľa zadania je potrebné mať nastavené nasledovné parametre konfigurácie zariadenia:

- Dátový bod pre aktuálnu teplotu
  - musí mať označenie (napr. **Teplota**) rovnaké ako v zdrojovom programe (viď nižšie)
  - nastavenú podmienku generovania užívateľskej udalosti na **Nová hodnota**
  - index užívateľskej udalosti z rozsahu 0..65535 nastavený používateľom (napr. 160 - pri nameraní novej hodnoty dátového bodu sa vygeneruje udalosť s nastaveným číslom 160)
  - správne nastavená horná hranica (podľa používateľských požiadaviek)
- Dátový bod pre indikáciu behu ventilátora
  - musí mať označenie (napr. **CoolRun**) rovnaké ako v zdrojovom programe (viď nižšie)
- Dátový bod pre indikáciu chyby chladenia
  - musí mať označenie (napr. **CoolErr**) rovnaké ako v zdrojovom programe (viď nižšie)
- Dátový bod pre ovládanie chodu ventilátora
  - musí mať označenie (napr. **Cooler**) rovnaké ako v zdrojovom programe (viď nižšie)

Popis riešenia
Meranie teploty prebieha podľa používateľom nakonfigurovanej časovej schémy pre analógový vstup (viď priložená konfigurácia na CD). Pri načítaní novej hodnoty sa vygeneruje udalosť. Obsluha tejto udalosti porovná, či aktuálna teplota je väčšia alebo rovná ako je nakonfigurovaná horná hranica tohto dátového bodu. Pri splnení podmienky sa nastaví hodnota dátového bodu **Cooler** na 1, čím dôjde k zapnutiu ventilátora. Zároveň sa tento stav poznačí do interného registra jazyka. Nakoniec sa spustí časovač, ktorý po 5 sekundách vyvolá udalosť na kontrolu činnosti chladenia. V prípade, že teplota nie je väčšia ako nakonfigurovaná horná hranica, nastaví sa hodnota dátového bodu **Cooler** na 0 (vypnutie ventilátora). Tento stav sa opäť poznačí do registra a spustí sa časovač na kontroli činnosti chladenia.
Po uplynutí časovača na kontrolu chladenia sa generuje udalosť, ktorá skontroluje, či poznačený stav v internom registri sa zhoduje s hodnotou dátového bodu **CoolRun** (indikácia, či je zapnuté/vypnuté chladenie). Ak je podmienka splnená, zapíše sa do dátového bodu **CoolErr** hodnota 0 (neindikuje sa chyba). V prípade nesplnenia podmienky sa indikuje chyba (hodnota 1 na dátový bod **CoolErr**) a zároveň sa znovu spustí časovač na ďalšiu kontrolu chladenia.
Program

```
// subor obsahujuci systemove konstanty zariadenia
#INCLUDE <system.spi>

// definicia konstant
#CONST Zapnut = 1
#CONST Vypnut = 0
```

```
// konstanty pre oznacenia datovych bodov v konfiguracii
#CONST Teplota = "Teplota"
#CONST Cooler = "Cooler"
#CONST CoolerRun = "CoolRun"
#CONST CoolerError = "CoolErr"

// konstanta pre pouzitie na odkaz do registrov
#CONST CoolerId = 0

// zakazanie debug informacii pre nasledujuce akcie
#DEBUG OFF
#ACTION AfterResetEvent Action_SYS Action_SYS_RST
// obsluha udalosti "Reset" zariadenia
#END

// povolenie debug informacii pre nasledujuce akcie
#DEBUG ON
// ID debug logu
#PARAM LogNumber 160
// maximalna velkost debug logu
#PARAM LogSize 4096
// obsluha udalosti novej hodnoty datoveho bodu
#ACTION NovaTeplota Action_DP_NEW 160
    // test, ci je hodnota datoveho bodu Teplota vacsia alebo rovna
    // ako horna hranica datoveho bodu Teplota
      GE FLOAT DP[Teplota], DP[Teplota].cfgUserHI
            // zapis hodnoty konstanty Zapnut na datovy bod
            MOV FLOAT DP[Cooler], Zapnut
            // zapis hodnoty konstanty Zapnut do registra interpretera W
            MOV W[CoolerId], Zapnut
            // nastavenie casovaca na 5 sekund
            MOV CTC[CoolerId], 5
      // ak bol test nepravdivy (Teplota je mensia ako horna hranica)
      ELSE
            // zapis hodnoty konstanty Vypnut na datovy bod
            MOV FLOAT DP[Cooler], Vypnut
            // zapis hodnoty konstanty Vypnut do registra interpretera W
            MOV W[CoolerId], Vypnut
            // nastavenie casovaca na 5 sekund
            MOV CTC[CoolerId], 5
      // ukoncenie podmienky (nemusi byt, lebo konci akcia)
      ENDIF
#END

#PARAM LogNumber CoolerId
#PARAM LogSize 2048
// obsluha udalosti po uplynuti casovaca CoolerId
#ACTION TestChladenia Action_CTC CoolerId
      // test, ci sa hodnota registra CoolerId rovna hodnote datoveho bodu
CoolerRun
      EQ FLOAT W[CoolerId], DP[CoolerRun]
            // zapis hodnoty konstanty Vypnut na datovy bod CoolerError
            MOV FLOAT DP[CoolerError], Vypnut
      // ak bol test nepravdivy
      ELSE
            // zapis hodnoty konstanty Zapnut na datovy bod CoolerError
            MOV FLOAT DP[CoolerError], Zapnut
            // nastavenie casovaca CoolerId na 1 sekundu
            MOV CTC[CoolerId], 1
      // ukoncenie podmienky (nemusi byt, lebo konci akcia)
      ENDIF
```

```
#END
```

Poznámky

Celý príklad aj s úplnou konfiguráciou je dostupný na inštalačnom CD aplikácie **K2config** v adresári **Examples\Ex1**.

12.2 Meranie hmotnosti

Zadanie

Na základe nameranej hmotnosti je potrebné nastaviť hodnotu prúdovej slučky 4-20 mA podľa nastavenia hardvérových hraníc snímača hmotnosti.

Popis technického riešenia

Meranie hmotnosti pomocou tenzometrického mostíka na analógovom vstupe 0.

Popis konfigurácie

V konfigurácii je potrebné mať správne nakonfigurované 2 dátové body podľa zadania a popisu technického riešenia. Pre správnu činnosť podľa zadania je potrebné mať nastavené nasledovné parametre konfigurácie zariadenia:

- Dátový bod pre aktuálnu hmotnosť
  - musí mať označenie (napr. **Hmotn**) rovnaké ako v zdrojovom programe (viď nižšie)
  - nastavenú podmienku generovania užívateľskej udalosti na **Nová hodnota**
  - index užívateľskej udalosti z rozsahu 0..65535 nastavený používateľom (napr. 150 - pri nameraní novej hodnoty dátového bodu sa vygeneruje udalosť s nastaveným číslom 150)
  - správne nastavená dolná hranica (podľa používateľských požiadaviek)
  - správne nastavená horná hranica (podľa používateľských požiadaviek)
- Dátový bod pre prepočítanú hodnotu hmotnosti
  - musí mať označenie (napr. **AOUT**) rovnaké ako v zdrojovom programe (viď nižšie)
  - správne nastavená dolná hranica (podľa používateľských požiadaviek)
  - správne nastavená horná hranica (podľa používateľských požiadaviek)

Popis riešenia

Meranie hmotnosti prebieha podľa používateľom nakonfigurovanej časovej schémy pre analógový vstup. Pri načítaní novej hodnoty sa vygeneruje udalosť. Obsluha tejto udalosti podľa nastavených dolných a horných hraníc dátových bodov **Hmotn** a **AOUT** pomocou lineárnej transformácie vypočíta novú hodnotu dátového bodu **AOUT**.

Program

```
// subor obsahujuci systemove konstanty zariadenia
#INCLUDE <system.spi>

// konstanty pre oznacenia datovych bodov v konfiguracii
#CONST Hmotnost = "Hmotn"
#CONST HmotnostOut = "AOUT"

// povolenie debug informacii pre vsetky nasledujuce akcie
#DEBUG  ON
// ID debug logu
#PARAM LogNumber 150
// maximalna velkost debug logu
#PARAM LogSize 4096
#ACTION HmotnostNova Action_DP_NEW 150
    // vypocet koeficientov pre linearnu transformaciu
    // k = (y2-y1) / (x2-x1)
      SUB  FLOAT W[0], DP[HmotnostOut].cfgHwMax, DP[HmotnostOut].cfgHwMin
      SUB  FLOAT W[1], DP[Hmotnost].cfgHwMax, DP[Hmotnost].cfgHwMin
      // k = W[0]
      DIV  FLOAT W[0], W[0], W[1]

      // q = y1 - k * x1
      MUL  FLOAT W[1], W[0], DP[Hmotnost].cfgHwMin
```

```
        // q = W[1]
        SUB   FLOAT W[1], DP[HmotnostOut].cfgHwMin, W[1]

        // y = k*x + q
        MUL   FLOAT W[2], W[0], DP[Hmotnost]
        ADD   FLOAT W[2], W[2], W[1]
        // na datovy bod sa zapise transformovana hodnota y = W[2]
        MOV   FLOAT DP[HmotnostOut], W[2]
#END
```

Poznámky

Celý príklad aj s úplnou konfiguráciou je dostupný na inštalačnom CD aplikácie **K2config** v adresári **Examples\Ex2**.

12.3 GSM brána

Zadanie

Umožniť rôznym používateľom ovládanie otvorenia/zatvorenia brány. Pri prezvonení nakonfigurovaným používateľom spustiť sekvenciu otvárania brány. Bránu je potrebné po 2 minútach automaticky zatvoriť, pokiaľ je ešte stále otvorená.

Popis technického riešenia

Ovládanie brány je pripojené na digitálny výstup 0. Koncový spínač zatvorenia brány je pripojený na digitálny vstup 0.

Popis konfigurácie

V konfigurácii je potrebné mať správne nakonfigurované 2 dátové body popisu technického riešenia. Pre správnu činnosť podľa zadania je potrebné mať nastavené nasledovné parametre konfigurácie zariadenia:

- Dátový bod koncového spínača zatvorenia brány
  - musí mať označenie (napr. **BranaSt**) rovnaké ako v zdrojovom programe (viď nižšie)
  - nastavenú podmienku generovania užívateľskej udalosti na **Zmenená hodnota**
  - index užívateľskej udalosti z rozsahu 0..65535 nastavený používateľom (napr. 200 - pri zistení zmeny hodnoty dátového bodu sa vygeneruje udalosť s nastaveným číslom 200)
- Dátový bod pre ovládanie brány
  - musí mať označenie (napr. **BranaImp**) rovnaké ako v zdrojovom programe (viď nižšie)

Popis riešenia

Sekvencia otvárania brány sa spúšťa pri výskyte udalosti prezvonenia a overenia používateľa. Obsluha nastaví impulz na dátovom bode **BranaImp** na 1 a spustí 1 sekundový časovač. Po uplynutí času, obsluha časovača nastaví impulz na dátovom bode **BranaImp** na 0.
Vždy pri zmene hodnoty dátového bodu **BranaSt** dôjde k overeniu, či hodnota dátového bodu **BranaSt** je rovná 0 (brána je otvorená). V prípade rovnosti sa spustí 2 minútové odpočítavanie (pomocou časovača). Po uplynutí daného času sa opäť na 1 sekundu vygeneruje impulz na dátovom bode **BranaImp**.

Program

```
// subor obsahujuci systemove konstanty zariadenia
#INCLUDE <system.spi>

#PARAM LogSize 16384

#CONST Opened = 0
#CONST Closed = 1

// konstanty pre oznacenia datovych bodov v konfiguracii
#CONST GateImpulse = "BranaIm"
#CONST GateState = "BranaSt"
```

```
// konstanty pre casovace
#CONST GateImpulseWait = 0
#CONST GateOpened = 1

// timeout ako dlho ma byt brana otvorena (v sek)
#CONST GateOpenedTimeout = 10

#ACTION Reset Action_SYS Action_SYS_RST
      MOV STS[tabEVT, evtSEL, selSPL], 1.0
#END

#DEBUG ON
// obsluha udalosti pri zavolani a overeni uzivatelovho cisla
#ACTION Pouzivatel Action_USR Action_USR_PHONEOK
    // nastavit impulz
      MOV FLOAT DP[GateImpulse], 1
      // spustit casovac GateImpulseWait
      MOV CTC[GateImpulseWait], 1
#END

// obsluha casovaca GateImpulseWait
#ACTION TimerGateOpen Action_CTC GateImpulseWait
      // zhodit impulz
      MOV FLOAT DP[GateImpulse], 0
#END

// obsluha udalosti zmenena hodnota datoveho bodu
#ACTION GateStateChanged Action_DP_SENSE 200
    // ak je brana otvorena
      EQ FLOAT DP[GateState], Opened
            // spustit 2 minutove odpocitavanie
            MOV CTC[GateOpened], GateOpenedTimeout
      // koniec podmienky (nie je potrebne)
      ENDIF
#END

// obsluha casovaca GateOpened
#ACTION TimerGateOpened Action_CTC GateOpened
    // ak je brana otvorena
      EQ FLOAT DP[GateState], Opened
            // nastavit impulz
            MOV FLOAT DP[GateImpulse], 1
            // spustit casovac GateImpulseWait
            MOV CTC[GateImpulseWait], 1
      // koniec podmienky (nie je potrebne)
      ENDIF
#END
```

Poznámky

Celý príklad aj s úplnou konfiguráciou je dostupný na inštalačnom CD aplikácie **K2config** v adresári **Examples\Ex3**.