RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境 CS+(CubeSuite+)版)、および RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境 High-performanceEmbedded Workshop 版) ご使用上のお願い

RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境 CS+(CubeSuite+)版)および RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境 High-performance Embedded Workshop 版) の使用上の注意事項をお知らせします。

- **初期値を伴う集成体に関する注意事項 (RXC#034)**

  注: 各注意事項の後ろの番号は、注意事項の識別番号です。

---

1. 該当製品およびバージョン

(1) RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境 CubeSuite+版) V1
型名: PRX00CSP1-MWR
CC-RX コンパイラ V1.02.00 ～ V1.02.01

(2) RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境 CS+(CubeSuite+)版) V2
型名 : PRX00CSP2-MWR
CC-RX コンパイラ V2.00.00 ～ V2.02.00

(3) RX ファミリ用 C/C++コンパイラパッケージ(統合開発環境 High-performance Embedded Workshop 版) V.1.00 Release 00 ～ V.1.02 Release 01
型名: PRX00CAS1-MWR
CC-RX コンパイラ V1.00.00 ～ V1.02.01

2. 内容

C99 および C++言語で、初期値がある集成体を自動変数として定義した時、
「初期化子の数 < 要素数」の場合は、初期値が設定されていない要素が
0 で初期化されず不定値となります。

3. 発生条件

以下の条件をすべて満たす場合に発生します。

(1) コンパイラオプションに-lang=c99 または-lang=cpp を指定している(注 1)。
(2) 初期値を持つ自動変数の集成体 (構造体または配列) の定義がある。
(3) (2)の集成体で初期化子の数が、要素数より少ない。
(4) (2)の集成体で初期化子に定数式 (注 2) がひとつもない。

注 1:
C99 および C++言語以外の場合に、初期値がある集成体を自動変数として定義した場合、
定数式以外の初期化子の記述は言語仕様違反となりますが、現状、コンパイルエラーは
発生せず、結果も保証されません。
また、C99 および C++言語以外の場合においても、上記(1)から(4)の発生条件を満たす場合は、
初期値が不定値となります。

注 2:
定数式は、演算式が定数値および静的変数のアドレスだけから構成されるものを意味します。

発生例 1: C99 言語の場合 - sample1.c

```
// コンパイルオプション: -cpu=rx600 -lang=c99   /* 発生条件(1) */
int get01a(void);
int get01b(void);
void check_ptr01(int *);
void func01(void)
{
      int array01[8] =           /* 発生条件(2) */
      { get01a(), get01b(), };   /* 発生条件(3)(4) */
                                 /* ここで、array[2]～array[7]が不定 */
                                 /* な値になります */
      check_ptr01(array01);
}
```

発生例 2: C++言語の場合 - sample2.cpp

```
// コンパイルオプション: -cpu=rx600 -lang=cpp   /* 発生条件(1) */
struct Str02
{
      short a02, b02, c02, d02;
};
short var02a;
short var02z;
void check_ptr01(int *);

void func01(void)
{
       struct Str02 st02 =          /* 発生条件(2) */
       { var02a };                  /* 発生条件(3)(4) */
                                    /* ここで、st02.b02, st02.c02, */
                                    /* st02.d02 が不定な値になります */
       var02z = st02.c02;
}
```

4. 回避策

以下のいずれかの方法で回避してください。

(1) 少なくとも一つ、0 となる定数式の初期化子を追加する。

(2) 集成体に初期値を設定せず、実行時に値を設定する。

3.の発生例 1 に対する回避策(1)の適用例

```
int get01a(void);
int get01b(void);
void check_ptr01(int *);
void func01(void)
{
       int array01[8] =
       { get01a(), get01b(), 0};    /* 0 となる初期化子を追加 */
                                    /* array[2]だけでなく、array[3]～ */
                                    /* array[7]も 0 に初期化される */
       check_ptr01(array01);
}
```

3.の発生例 2 に対する回避策(2)の適用例:

```
#include                  /* memset を用いるため */
struct Str02
{
        short a02, b02, c02, d02;
};
short var02a;
short var02z;
void check_ptr01(int *);
void func01(void) {
        struct Str02 st02;           /* 定義で初期値を設定しない*/
      memset(&st02,0,sizeof(st02));  /* st02 全体を 0 初期化する */
        st02.a02 = var02a;           /* 代入により初期化する */
        var02z = st02.c02;
}
```

5. 恒久対策

(1) RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境 CubeSuite+版) V1
および RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境
High-performance Embedded Workshop 版)

改修予定はありません。回避策を適用してください。

(2) RX ファミリ用 C/C++コンパイラパッケージ (統合開発環境 CS+(CubeSuite+)版) V2

次期バージョンで改修する予定です。

株式会社 日立超LSIシステムズ