

ルネサス提供ビットフィールド宣言「`iodefine.h`」を

デバッガ「AH8000」で可視化する方法

1. 対象品種

- ・CC-RX (ELF/Dwarf2 or 3) でコンパイル/リンクしたRX品種が対象。
- ・GCC for RenesasRX (ELF/Dwarf2 or 3 or 4) でコンパイル/リンクしたRX品種が対象。

2. 概要説明

- ・ルネサス提供の「`iodefine.h`」を利用してプログラムした場合に、ビットフィールド情報を可視化する方法を説明する。

3. 実装準備

| 品種 | 「 <code>iodefine.h</code> 」をベースにした構造体ポインタ宣言 |
|-----------|---|
| RX230/231 | <code>Iostruct_rx230.c</code> |
| RX23T | <code>Iostruct_rx23t.c</code> |
| RX631/63N | <code>Iostruct_rx63n.c</code> |
| RX64M | <code>Iostruct_rx64m.c</code> |
| RX651/65N | <code>Iostruct_rx65x.c</code> |
| RX66N | <code>Iostruct_rx66n.c</code> |
| RX72M | <code>Iostruct_rx72m.c</code> |

```

<Iostruct_rx65x.c> ソース内容 (一部)
#include "iodefine.h" // Include iodefine.h
volatile struct st_bsc * const _BSC = {(struct st_bsc *)&BSC};
volatile struct st_cac * const _CAC = {(struct st_cac *)&CAC};
volatile struct st_can * const _CAN0 = {(struct st_can *)&CAN0};
volatile struct st_can * const _CAN1 = {(struct st_can *)&CAN1};
volatile struct st_cmt * const _CMT = {(struct st_cmt *)&CMT};
volatile struct st_cmt0 * const _CMT0 = {(struct st_cmt0 *)&CMT0};
  
```

<準備>

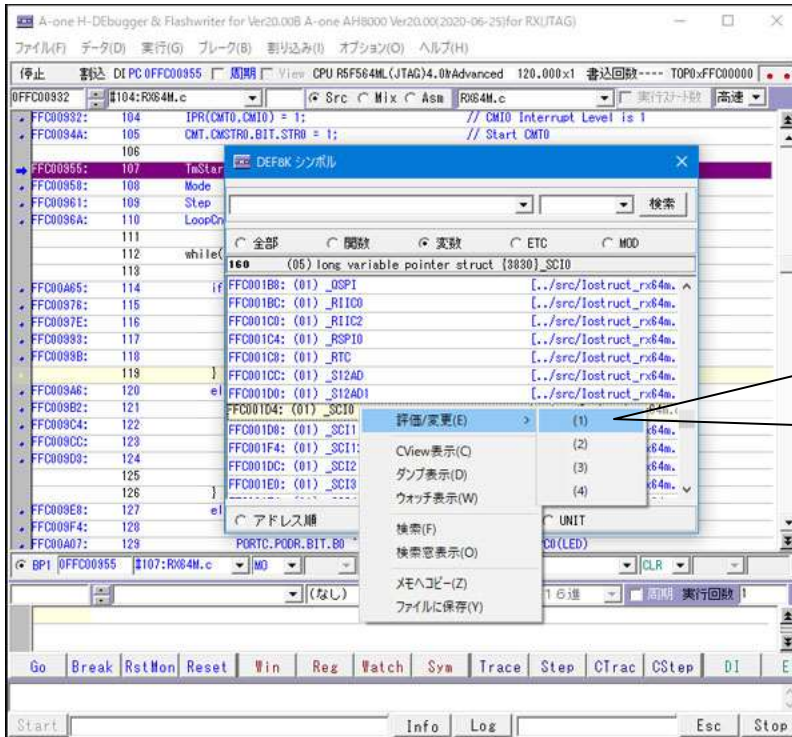
- (1) 上記のRX品種別の「`iodefine.h`」をベースにした構造体ポインタ宣言のCソースを下記URLからダウンロードする。

<https://aone.co.jp/tools/AH8000/Ctext/index.html>

- (2) ダウンロードしたCソースをプロジェクトに実装してビルドする。なお、このCソースは、各構造体ポインタのアドレス値をROMテーブル化するだけの宣言のみのソースとなります。

4. 操作例

<4-1>



評価/変更 (1) を選択します。

操作例「SCIO」で示す。

<4-2>



[SIO0]は、ROM エリア[0xFFFF801C4]に配置され、IO アドレス値[0x0008A000]が確認できます。

マウスをあてダブルクリックします。

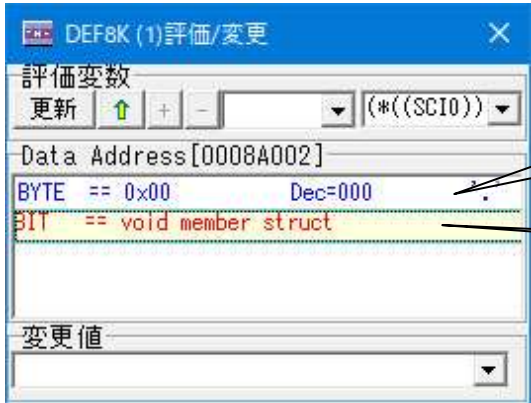
<4-3>



[SCIO]のメンバ宣言「struct st_sci0」が確認できます。

[SCR]にマウスをあてダブルクリックします。

<4-4>




The screenshot shows the 'DEF8K (1) 評価/変更' window. The '評価変数' section has '更新' set to '↑' and '*(SCI0)'. The 'Data Address' is '0008A002'. The 'BYTE == 0x00 Dec=000' line is visible. The 'BIT == void member struct' line is highlighted in yellow. Callout boxes indicate that the union member can be confirmed and that a double-click on the BIT line is required.

[SCR]のユニオンメンバが確認できます。

[BIT]にマウスをあてダブルクリックします。

<4-5>



The screenshot shows the 'DEF8K (1) 評価/変更' window with the same settings as in <4-4>. The 'BIT == void member struct' line is now expanded to show a list of bit fields: TIE, RIE, TE, RE, MPIE, TEIE, and CKE. Each field is followed by its bit mask in hexadecimal and decimal notation. Callout boxes indicate that the bit field definition for [SCI0->SCR.BIT] can be confirmed and that the bit position and content can be verified.

ビットフィールドが[SCI0->SCR.BIT]のビット構成が確認できます。

ビット位置と内容が確認できます。

以上でビットフィールドの参照例の説明を終了します。

5. 注意事項

- 本文書の著作権は、エーワン（株）が保有します。
- 本文書を無断での転載は一切禁止します。
- 本文書に記載されている内容についての質問やサポートはお受けすることが出来ません。
- 本文章に関して、ルネサス エレクトロニクス社への問い合わせは御遠慮願います。
- 本文書の内容に従い、使用した結果、損害が発生しても、弊社では一切の責任を負わないものとします。
- 本文書の内容に関して、万全を期して作成しましたが、ご不審な点、誤りなどの点がありましたら弊社までご連絡くだされば幸いです。
- 本文書の内容は、予告なしに変更されることがあります。