≫ 文法の曖昧さを理解して確実性と再利用性を高める

マイコンC言語 転ばぬ先のつえ

第9回　派生型④…移植性は無いが可読性バツグン！ビット・フィールド

鹿取 祐二

C言語は，誕生から50年近く経つ今でも，分野を問わず，さまざまな場面で使われるプログラミング言語です．言語仕様（文法）は標準規格化されていますが，曖昧な部分が多く存在します．

本連載では，C言語の文法の曖昧な部分と，それにより起こる問題を解説します．再利用性と効率が高く，安全かつ安心して使えるソフトウェアが開発できるようになることを目指します．

今回は前回（本誌2021年10月号）に続いて派生型，いわゆる基本型から派生した配列，ポインタ，構造体，共用体，ビット・フィールドについて紹介します．各派生型が移植性を重視した組み込み系システムで利用できるものなのかを理解しましょう．　＜編集部＞

15 ビット・フィールドは移植性がなく文法も曖昧だが…周辺機能の制御に向く

■ 概要

● ビット単位でデータを扱えるが問題点もある

▶周辺機能制御に向くが…移植性はない

ビット・フィールドは，他のビットを考慮せず，目的のビットだけを操作できる機能です．

筆者が1993年に本誌でビット・フィールドを周辺機能制御に応用する方法を紹介してから，瞬く間に利用が広がりました．今では当たり前のように組み込み系でビット・フィールドが使われるようになっています．

事実，ビット・フィールドを使えば周辺機能制御のプログラムは非常に分かりやすく，可読性は格段に向上します．その一方で，移植性がないという厳しい意見も多くいただきました．

▶文法が曖昧…MISRA-Cでも厳しいルールを制定

確かに，ビット・フィールドの文法には曖昧な部分が非常に多くあります．MISRA-Cでもビット・フィールドに関しては非常に厳しいルールが制定されていて，それらを読むと使う気をなくしてしまいます．つまり，ビット・フィールドはもろ刃の剣です．プログラミングの容易さや可読性を取るか，移植性を取るかをてんびんに掛けなければなりません．

▶本稿で解説すること

本連載では，ビット・フィールドを使うかどうかの判断は，読者の皆さんの判断に委ねたいと考えています．そこで，その判断材料とするために，今回はビット・フィールドの文法の何が曖昧なのかを明確にします．それが分かればMISRA-Cのビット・フィールドに対するルールも理解できると思います．

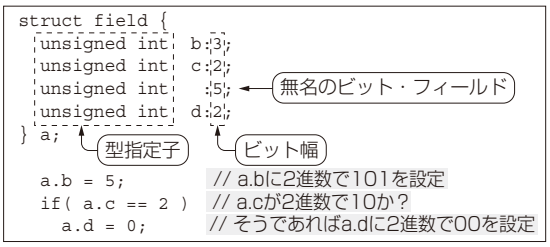
● 使い方

図1に示すのは，ビット・フィールドの基本的なコーディング例です．

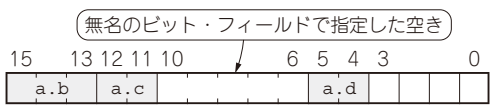
▶ビット・フィールドの記述方法

ビット・フィールドの記述は，構造体とほぼ同じです．struct，タグ名，中括弧の中はメンバの宣言，必要であれば最後に変数名を記述します．構造体との違いは，中括弧のメンバの宣言です．ビット・フィールドの場合は，メンバ名の後に「:」を記述し，その後にビット幅を整数定数で指定します．

図1（a）であれば，unsigned int型の先頭3ビットにb，次の2ビットにcというメンバ名を付けます．



（a）ソースコード



（b）ビット構造

図1　ビット・フィールドの使い方

基本的なコーディング例．他のビットを考慮せず，目的のビットだけを操作できることが特徴である．ソースコードの記述は構造体とほぼ同じ．ここでint型は16ビットで記載している

第1回　整数型①…値の範囲と負の内部表現（2021年1月号）
第2回　整数型②…表現できない値を代入したときの規則（2021年2月号）
第3回　整数型③…整数の格上げと算術変換規則（2021年3月号）