

MAXQユーティリティROMに 搭載された関数へのアクセス

MAXQユーティリティROMには、プログラム空間に保存されたデータやテーブルにアクセスするルーチンが用意されています。

マイクロコントローラのプログラミングでは、通常アプリケーションコード側にルックアップテーブルを用意します。シングルサイクルを特徴とするMAXQコアでは、アプリケーションからコード空間を直接読み出すことは不可能で、アプリケーションコード内にテーブルを用意しても直接アクセスすることはできません。その代わりに、MAXQユーティリティROMには、プログラム空間に保存されたデータやテーブルにアクセスするルーチンが用意されています。これらはすべてのROMに搭載されているコア関数であり、各MAXQのROMには、他のルーチンも搭載されている場合があります。各関数の位置はROM内で一定しておらず、ROMのバージョンによっても異なる可能性があるため、これらのルーチンに対する標準的なアクセス方法が定められています。この方法を使うと、あるバージョンのROMを使って書いたコードが、その後ROMのバージョンに変更があっても、書き直したりコンパイルし直すことなく使用することができます。

さまざまなMAXQプロセッサには様々な種類がありますが、それぞれのユーティリティROMには、各プロセッサがサポートする関数のアドレステーブルがあります。テーブルの記録位置は部分によって異なる可能性があるため、テーブルに対するポインタをアドレス800Dhに記録することになっています。各関数のアドレスは、テーブルのインデックスによって見つけることができます。テーブルに関数を記述する順番は、あるROMについては、バージョンにかかわらず一定です。MAXQ2000の関数とテーブル内のエントリーポイントを表1に示します。

表1. MAXQ2000ユーティリティROMユーザ関数テーブル

FUNCTION NUMBER	FUNCTION NAME	ENTRY POINT (USERTABLE = ROM[800Dh])
0	Reserved	ROM[userTable + 0]
1	Reserved	ROM[userTable + 1]
2	Reserved	ROM[userTable + 2]
3	moveDP0	ROM[userTable + 3]
4	moveDP0inc	ROM[userTable + 4]
5	moveDP0dec	ROM[userTable + 5]
6	moveDP1	ROM[userTable + 6]
7	moveDP1inc	ROM[userTable + 7]
8	moveDP1dec	ROM[userTable + 8]
9	moveFP	ROM[userTable + 9]
10	moveFPinc	ROM[userTable + 10]
11	moveFPdec	ROM[userTable + 11]
12	copyBuffer	ROM[userTable + 12]

ユーティリティROM関数の実行は、4つのステップで行います。最初のステップでは、関数テーブルの位置をアドレス800Dhから取得します。2番目のステップとして、この位置に、実行したい関数のオフセットを加えます。3番目のステップでは、算出した位置からユーティリティ関数のアドレスを取得します。最後のステップとして、テーブルから取得した位置を呼び出し、目的の機能を実行します。4つのステップによる実行の例として、MAXQ2000のmoveDP1inc関数をMAXQアセンブリ言語で実行するコードを示します。

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Function:      ReadDataAtDP1
;; Description:   This function uses the utility ROM function "moveDP1inc"
;;               to read from program memory the data stored at the
;;               address in DP[1].  If DP[1] is in word mode two
;;               bytes will be read.  If DP[1] is in byte mode only
;;               one byte is read.  DP[1] is then post incremented.
;; Returns:      The result is returned in GR.
;; Destroys:     ACC and DP[0]
;; Notes:        This function assumes that DP[0] is set to word
;;               mode and the device has 16-bit accumulators.

```

```

ReadDataAtDP1:
    move DP[0], #0800Dh ; This is where the address of the table is stored.
    move ACC, @DP[0]   ; Get the location of the function table.
    add #7              ; Add the index to the moveDP1inc function.
    move DP[0], ACC    ; Point to where the address of moveDP1 is stored.
    move ACC, @DP[0]   ; Retrieve the address of the function.
    call ACC           ; Execute the function.
    ret

```

MAXQ ROMがバージョンアップするとユーティリティ関数の保存位置が変更されるかもしれませんが、ReadDataAtDP1関数を呼び出す例と同じ形式のルーチンとしておけば、前方互換性が保証されます。ただし、互換性の代償として、コードサイズが大きくなり、実行時間も長くなります。これでは犠牲が過大となり、ユーティリティROM関数を直接呼び出す方がいいと考えられるケースもあるでしょう。そのような場合には、実行したい関数の位置を求めておき、その位置をcallのデスティネーションとして使うことも可能です。

ユーティリティ関数を使用する状況としてよくある操作として、コード空間に書き込まれた文字列の読出しが考えられます。プログラマが用意する可能性がある文字列としては、アプリケーション実行中に表示するエラー用文字列や各種情報、デバッグ用文字列などがあるでしょう。次のコードセグメントに示すのは、前述の方法でReadDataAtDP1を呼び出し、このような文字列を読み出す例です。

```

Text:
    DB "Hello World!",0 ; Define a string in code space.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Function:      PrintText
;; Description:   Prints the string stored at the "Text" label.
;; Returns:      N/A
;; Destroys:     ACC, DP[1], DP[0], and GR.
;; Notes:        This function assumes that DP[0] is set to word mode,
;;               DP[1] is in byte mode, and the device has 16-bit
;;               accumulators.

```

```

PrintText:
    move DP[1], #Text ; Point to the string to display.
    move ACC, DP[1]   ; "Text" is a word address and we need a
    sla              ; byte address, so shift left 1 bit.
    or #08000h       ; Code space is mapped to 8000h when running
    move DP[1], ACC  ; from the ROM, so the address must be masked.
PrintText_Loop:
    call ReadDataAtDP1 ; Fetch the byte from code space.
    move ACC, GR
    jump Z, PrintText_Done ; Reached the null terminator.
    call PrintChar ; Call a routine to output the char in ACC
    jump PrintText_Loop ; Process the next byte.
PrintText_Done:
    ret

```

ユーティリティROM
ルーチンに対する標準的な
アクセス方法も用意されて
おり、あるMAXQプロセッサ
について、バージョンに
かかわらず動作するコード
を書くことができます。

まとめ

ユーティリティ関数を使うと、プログラムメモリに記録されたデータを簡単に読み出すことができます。ユーティリティROMルーチンに対する標準的なアクセス方法も用意されており、あるMAXQプロセッサについて、バージョンにかかわらず動作するコードを書くことができます。ライブラリは、一度、構築すれば再利用することができます。将来、ROMバージョンによって互換性が失われるかもしれないという心配は必要ありません。