K-008

教育向けμITRON 仕様準拠 OS の 8 ビット PIC への実装

Implementation of education-oriented micro-ITRON specification on 8-bit PICs

秋葉 将和 †　堀田 忠義 †　寺内 美奈†　菅野 恒雄‡
Masakazu Akiba　Tadayoshi Horita　Mina Terauchi　Tsuneo Kanno

１．はじめに

8 ビット PIC マイコンを使用した授業案や授業実践例は，中等教育，高専，職業訓練校，大学など幅広い教育現場から多数報告されている[1][2][3][4]．PIC マイコンが教育現場において広く使用される要因としては，安価で入手が容易，多様な種類，書籍やインターネット上での情報が豊富な点などの他に，DIP タイプ・5V 電源対応等，授業実践上の利便性の高さも一因であると考えられる．

一方，産業界においては組込み機器の多機能化に伴い，組込みソフトウェアの大規模化が進んでいる．このため，多くのプロジェクトではソフトウェア開発の効率化を目的として，組込みオペレーティングシステム(以下，組込み OS)を使用している．使用された組込み OS の種類としては，2010 年版の経済産業省の国内調査結果[5]によると ITRON 仕様 OS が最も多かった．加えて同調査によると，「組込みソフトウェア開発の課題解決の有効な手段」の回答の中では，「技術者のスキル向上」が最も多い．これらの事から，今後も ITRON 仕様 OS に関わる教育を含めた組込みソフトウェア技術者の人材育成は，重要であると言える．

しかし，教育現場おける PIC マイコンを用いた授業実践例では，ほとんどの場合，周辺入出力回路の実装とその制御を行う比較的単純なプログラム開発を想定している．OS を搭載するケースはほとんど見られない．PIC マイコンで動作可能な組込み OS は非常に限られるため，ITRON 仕様 OS の実験実習を行いたい場合は，対応するマイコンボードを別途用意せざるを得ない．

そこで，本研究では 8 ビット PIC マイコンで動作可能なμITRON 仕様[6]準拠 OS の開発を行った．これによって，PIC マイコンの利便性を生かしたコンピュータ制御の初歩的実習から，国内で最も多く用いられている組込みリアルタイム OS である ITRON 仕様 OS まで，同一環境で段階的に学習可能となった．本研究で開発したμITRON 仕様準拠 OS は，教育性を重視し，マルチタスク動作を理解するためのトレースログ出力やタスク状態表示などの機能を持つ．以下，このμITRON 仕様準拠 OS の実装及び機能特徴について述べ，職業能力開発総合大学校（以下，本学）電子情報専攻における授業実践例について報告する．

２．ターゲット PIC マイコン

8 ビット PIC マイコンの中でもミッドレンジの PIC16 シリーズは入門書で多く取り上げられ，教育現場でもよく用いられるマイコンである．しかし，メモリサイズや関数戻り番地を保持するハードウェアスタック（以下，HW スタック）の段数が非常に限られるため，カーネル実装には制約が大きい．そこで，DIP タイプ，5V 電源，ROM サイズが 32KB 以上，RAM サイズが 2KB 以上，構造化に十分な HW スタック段数を要件として検討した結果，PIC18F シリーズをターゲットとして選定した．例えば，PIC18F2620，PIC18F4620，PIC18F26k22，PIC18F46K22 などの PIC18F シリーズマイコンは，64KB ROM (Flash)，3968B RAM を搭載しており，数百円程度で購入可能である．これらの PIC マイコンにはタイマ，AD 変換，コンパレータ，USART，I2C，SPI などの機能モジュールが内蔵されており，Microchip Technology 社から無償提供されている関数によって手軽に利用することができる．これらの機能を応用したコンピュータ制御の実験・実習用として有用である．

†職業能力開発総合大学校
‡新潟職業能力開発短期大学校

３．開発環境

本カーネルの開発には，Microchip Technology 社から無償で提供されている統合開発環境 MPLAB IDE を用いた．また，C コンパイラとしては MPLAB C18 Lite 版を用いた．本カーネル上で動作するユーザアプリケーションの開発も同様の環境を使用する．MPLAB IDE には多機能で利便性の高いシミュレータ・デバッガが付属しており，カーネル開発時のみならず，ユーザアプリケーション開発時のマルチタスク動作解析にも有用である．これらの開発環境は，フリーソフトウェアのみで構築可能であり，これは，教育現場での利用のしやすさを考慮すると重要な要件である．

４．実装カーネルの機能と特徴

本カーネルではμITRON4.0 仕様のスタンダードプロファイルの中で，タスク管理機能，タスク付属同期機能，同期・通信機能，時間管理機能（システム時刻管理，周期ハンドラ），システム状態管理機能，割込み管理機能を実装している（一部未実装含む）．また，スタンダードプロファイルには含まれないが教育用途での重要性を考慮し，状態参照系のサービスコール(ref_yyy)，非タスクコンテキスト用の拡張サービスコール(ixxx_yyy)などを含め現段階で合計 76 のサービスコールと，独自仕様も含めた 13 の静的 API，静的 API を処理するコンフィギュレータから構成されている．カーネルのソースコード行数は空白行とコメント行を除き約五千行ほどであり，ほとんどの部分を C 言語にて記述している．ただし，複雑なスタック操作と処理速度が要求されるディスパッチ部分は，一部アセンブラにて記述している．

以下に本カーネル実装と機能特徴について述べる．

４．１　タスクスタックの実現方法

今回のカーネル実装においては，いかにメモリ使用量を削減するかが最も大きな課題となる．通常，マルチタ

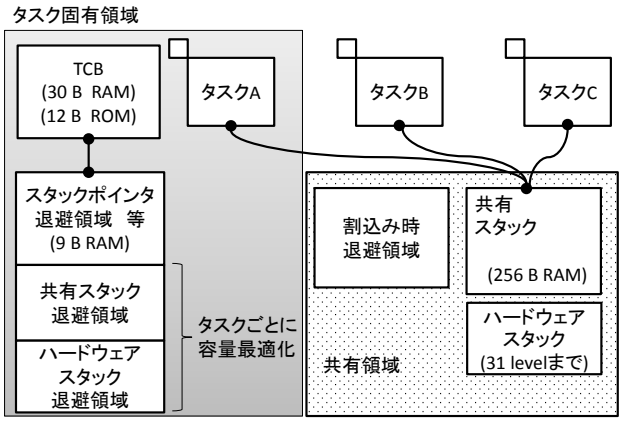

図 1　タスク共有領域と固有領域

スク環境ではタスクごとにスタック領域を確保する必要があり，これがメモリ使用量が増える大きな要因となる．本システムは教育用途での使用を主目的としており，ログ出力や変数値確認などのためにスタック消費の大きい printf 系関数の多用も想定している．ターゲット PIC の RAM 容量は数 KB であり，タスクごとに独立したスタック領域を確保した場合，生成できるタスク数は限られる．

そこで，本カーネルでは全てのタスクが同一のスタック領域を共有して使用し，必要に応じて予め確保してある退避領域に退避する構造を採用している．これは本カーネル実装上の最も特徴的な点といえる．通常，複数のタスクがスタック領域を共有する場合，これらのタスクは「待ち状態への遷移不可」，「優先度を変更不可」など機能に制限が設けられる．しかし，本カーネルでは共有スタック構造を用いながら，タスクに機能的制限はなく，待ち状態への遷移やタスク切り替え動作を行う．このため，タスク切り替え処理においては，中断タスクのスタックデータ(共有スタック，HW スタック)を別の領域へ退避し，タスク再開時にこれらを復帰する必要がある．

本カーネルにおけるスタック領域の概要を図 1 に示す．共有スタックのサイズは 256B としている．実行中タスクの Auto 変数や関数引数はこの共有スタックに積み上げられる．また，呼び出し関数の戻り番地は HW スタックに積み上げられる．タスクごとの固有領域としては，RAM 上に置くタスク制御ブロック(TCB:Task Control Block)が 30B，ROM 上に置くことが可能なタスク初期化ブロックが 12B となっている．さらにこれに加えて，図 1 に示すようにスタック(共有スタック，HW スタック)退避領域が必要となる．このタスクごとのスタック退避領域は，当該タスクからのサービスコール発行に伴うタスク切り替え時のスタック退避先として使用する．サービスコール発行に伴うタスク切り替えに限定した場合，退避対象は「サービスコール発行時」のスタックデータとスタックポインタ・フレームポインタのみである．従って，この退避領域のサイズは，当該タスクの動作に必要なスタックサイズ全量に比べ大幅に削減できる場合が多い．また，タスクのコードを静的に解析することによって必要最小限の退避領域サイズを予め算出できる．本システムではコンフィギュレータに静的コード解析による退避領域サイズの自動算出機能を付加し，ユーザがこれらの内部実装を意識せず実験・実習できる環境を提供している．

一方，割込みによって実行タスクが中断され，より優先度の高いタスクへの切り替えが発生する場合，中断タスクのスタック(共有スタック，HW スタック)に加えてレジスタ類(15B)とワークエリア(29B)の退避が必要となる．この割込み時用の退避領域は，タスクごとではなく共用として別途確保する．退避領域サイズと領域個数はユーザがコンフィギュレーションファイルで指定する．ただし，割込みが入るタイミングは予測できないため，必要最低限の領域サイズ・領域個数を正確に見積もることは難しい．そこで，本カーネルは「割込み時退避領域」の領域不足を検知した場合，システムを停止し，ユーザにその旨を表示する．これによって，ユーザは試行錯誤的に「割込み時退避領域」の必要サイズを決定できる．

以上に述べたスタック実現方法は，生成タスク数が増えるほど RAM 節約の効果が大きい．反面，スタックの退避・復帰処理がタスク切り替え時のオーバーヘッドとなる．スタック退避・復帰処理がタスク切り替え時間に与える影響については後述する．

４．２　ログ・デバッグ機能

本カーネルのマルチタスクプログラミング教育用途での使用を考慮した場合，タスク動作解析のためのログ機能や状態表示機能は重要な機能である．本カーネルではサービスコール発行履歴表示，タスク状態一覧表示，セマフォ・イベントフラグ・データキュー・メールボックス状態表示，レディキューや各種キューデータ表示,タスク毎のディスパッチ回数統計表示などのデバッグ用関数を実装している．これらの機能が不要な場合は，カーネルヘッダファイル内のマクロ定義にて機能を無効にし，カーネルを軽量化できる．

また，MPLAB IDE のシミュレータ・デバッガ機能を用いることによって，シミュレータ上でタスクの動作解析が可能である．このシミュレータにはブレークポイントの設定，ステップ動作，トレース機能，変数のウォッチ，疑似入出力，ロジックアナライザ，実行時間の計測などの機能があり，カーネルやサービスコールの動作を理解するためのツールとして有用である．

４．３　メモリサイズ

カーネルのログ・デバッグ機能を無効にした場合，約 12KB ROM/430B RAM がカーネル領域として必要となる．ログ・デバッグ機能を有効にした場合は，約 20KB ROM/630B RAM がカーネル領域として必要となる．サービスコールはライブラリ化されており，ユーザアプリケーションに必要なサービスコールのみがリンクされる．また，前述した通り各タスクについて TCB とスタック退避領域が必要となる．これに加えて「割込み時退避領域」をコンフィギュレーションファイルにて必要に応じて確保する．

４．４　タスク切り替え時間

本実験では，MPLAB IDE 付属のシミュレータの実行時間計測機能を用いて，サービスコール発行に伴うタスク切り替え時間を計測した．使用デバイスは PIC18F46K22（動作クロック 64MHz）とし，本カーネルのログ・デバッグ機能は無効状態とした．本カーネルのタスク切り替え時間は，スタック退避・復帰処理の有無，及び，そのデータ量に依存する．実行中タスクが「休止状態」へ遷移し，次タスクが先頭から「開始」となる場合，スタッ

表 1　タスク切り替え時間

スタック退避処理	スタック復帰処理	最小タスク切り替え時間(μs)
無	無	13.2
無	有	23.9
有	無	32.0
有	有	43.2

図 2　PIC 実験用ボードの外観

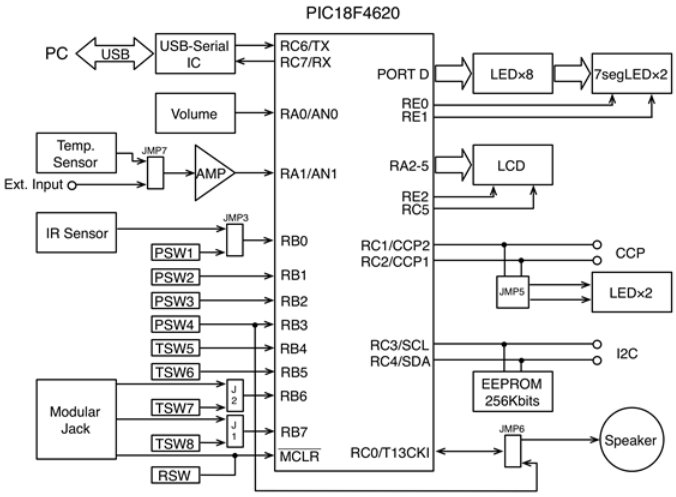

図 3　PIC 実験用ボード構成図

ク退避・復帰処理はいずれも発生しない．実行中タスクが「中断」となる場合はスタック退避処理が発生し，次タスクが中断位置から「再開」となる場合はスタック復帰処理が発生する．表 1 にスタック退避・復帰処理の有無によるタスク切り替え最少時間を示す（サービスコール毎の固有の処理時間は含めていない）．この時間に加えて，退避（または復帰）する共有スタックサイズ 1B 当たり約 0.3μ秒，HW スタック 1 レベル(2B)当たり約 0.6μ秒の処理時間増加となる（64MHz 動作時）．

タスク切り替え時間が増減する特性は，リアルタイム OS の特性として本来好ましくない．しかし，タスク切り替え時間は見積もることが可能であり，大きなサイズの Auto 変数や極端な関数ネスト構造を避けるといった基本的なプログラミングルールによってスタック退避・復帰時間の増加は抑制できる．

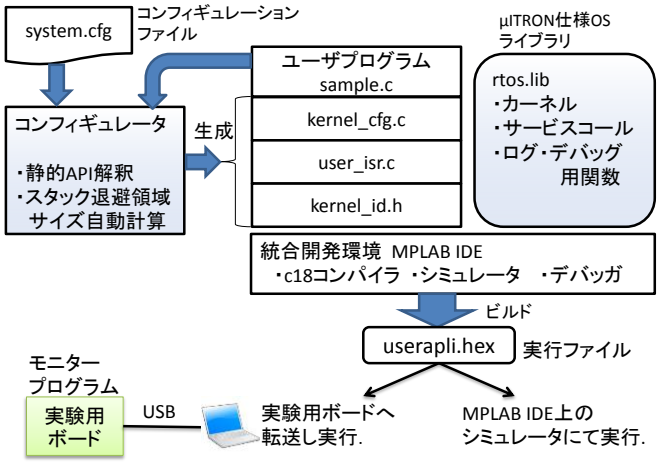

図 4　実習環境

５．　実習環境

本システムを本学電子情報専攻実習科目に導入し，リアルタイム OS の実験・実習を行っている．以下に実習機器及びユーザプログラム開発環境について述べる．

５．１　実験用ボード

実装したμITRON 仕様 OS は, MPLAB IDE に付属するシミュレータのみでタスク動作を確認できるが，実際の制御対象があった方が教育効果は大きい．そこで，図 2 に示す PIC18F4620 を用いた実験用ボードを使用している．これは本学電子情報専攻 1 年次のリテラシー教育として製作しているボードである[7]．2 年次の「組込みソフトウェア実習」では，1 年次に学生自身が製作した実験用ボードを用いてμITRON の実験・実習を行っている．図 3 に実験用ボードの構成図を示す．

５．２　ユーザプログラム開発環境

図 4 に実習プログラムの開発環境を示す．ユーザは，コンフィギュレーションファイル(system.cfg)とユーザプログラムのソースコードを記述する．本研究で実装したコンフィギュレータはこれらを読み込み，静的 API の解釈，ソースコード静的解析によるスタック退避領域サイズの自動算出を行い，構成・初期化ファイル(kernel_cfg.c)，ID 自動割り付けヘッダファイル(kernel_id.h)，ユーザ定義割込みハンドラ構成ファイル(user_isr.c)を生成する．これらと本カーネルライブラリ(rtos.lib)をビルド・リンクし，実行ファイル(userapli.hex)を得る．

動作確認は，シミュレータまたは実験用ボードで行う．シミュレータ上の実行では，ブレークポイントの設定，ステップ動作，カーネル内部構造体(TCB，セマフォ，イベントフラグなどの管理情報)のウォッチ，実行時間計測，任意位置での割込み発生シミュレートなどを通して，サービスコール動作やタスクスケジューリングを解析する多様な実習課題を設定できる．また，実験用ボードでは入出力デバイスを活用したマルチタスク制御プログラミング実習が可能である．実験用ボードの PIC には，USB を介したシリアル接続にて，実行ファイルダウンロードとプログラム起動が可能な自作モニタープログラムが書き込まれている．これによって，実験用ボードへのプロ

図 5　デモプログラム動作例

```
COM8:115200baud - Tera Term VT
ファイル(F)  編集(E)  設定(S)  コントロール(O)  ウィンドウ(W)  ヘルプ(H)
RTOS:>d

Task           PRI       dly_Time    act  wup  sus    save          Soft    Task
 ID  stat wait Now[Set]  Now[ Set ] cnt  cnt  cnt    flag   STKP    Stack   Name
  0  RDY  CLR   6[ 6]      0[    0]   0    0    0      0    0[ 0]    0[  0]  Idle
  1  WAI  SLP   1[ 1]      0[    0]   0    0    0      1    3[ 3]   30[ 33]  Clock
  2  WAI  DLY   1[ 1]     48[   50]   0    0    0      1    3[ 3]   13[ 18]  read_
  3  WAI  DLY   1[ 1]    197[  300]   0    0    0      1    3[ 3]   15[ 20]  read_
  4  RUN  CLR   5[ 5]      0[    0]   0    0    0      0    3[ 3]   32[ 32]  vol_D
  5  WAI  MBX   5[ 5]      0[    0]   0    0    0      1    3[ 3]   36[ 36]  temp_
  6  WAI  DLY   2[ 2]    401[  811]   0    0    0      1    3[ 3]   16[ 19]  LED_C
  7  WAI  SLP   3[ 3]      0[    0]   0    0    0      1    3[ 3]   23[ 26]  PSW1_
  8  DMT  CLR   3[ 3]      0[    0]   0    0    0      0    0[ 3]    0[ 13]  LED10
  9  WAI  SLP   3[ 3]      0[    0]   0    0    0      1    3[ 3]    8[ 12]  Sound
 10  WAI  SLP   3[ 3]      0[    0]   0    0    0      1    3[ 3]   23[ 26]  PSW2_
 11  WAI  SLP   1[ 1]      0[    0]   0    0    0      1    3[ 3]   24[ 27]  Count
 12  WAI  SLP   1[ 1]      0[    0]   0    0    0      1    3[ 3]    9[ 16]  Count
 13  DMT  CLR   4[ 4]      0[    0]   0    0    0      0    0[ 3]    0[ 25]  Displ
 14  WAI  FLG   1[ 1]      0[    0]   0    0    0      1    3[ 3]   21[ 21]  Task_
```

図 6　タスク状態表示（実験用ボードでの動作）

グラム書込みのための PIC ライタ等の追加装置は不要となる．

以上より，これらの実習環境では，フリーソフトウェアを使用し，PIC ライタが不要で，かつシミュレータのみでも実習可能であるため，受講者の自宅での低コスト自学自習を可能にしている．

５．３　ユーザプログラムの例

本システムと実験用ボードを用いたユーザプログラムの動作例を図 5 に示す．これは，実験用ボードのマルチタスク制御デモとして作成したサンプルプログラムである．主な機能は，時刻表示，温度センサ値読み取り，ボリューム値読み取り，LED 点滅，カウンタ，7 セグメント LED ダイナミック点灯，メロディ出力，トグルスイッチによる機能切り替えなどであり，これらは同時並行的に動作する．図 6 は，実験用ボードにてこのプログラムを実行した際のタスク状態表示関数の出力例である．実験用ボードは，USB にてシリアル接続した PC からのコマンドを受け付け，コマンド受信タイミングにおけるタスクステータス（ID，状態，待ち要因，優先度，遅延時間，要求キューイング数など）を出力している．このサンプルプログラムでは，14 個のタスク，4 個のユーザ定義割込みハンドラ，2 個の周期ハンドラ，4 個のメールボックス，2 個のデータキュー，1 個のイベントフラグ，1 個のセマフォを使用している．本カーネルのログ・デバッグ機能を有効にしてビルドした場合，メモリ使用量は 49KB ROM/2.4KB RAM ほどとなった．図 7 にこのサンプルプログラム開発時の画面例(MPLAB IDE)を示す．　図 7 では，シミュレータを用いてプログラムを実行し，図 6 の実機動作同様のタスク状態表示出力を行っている．

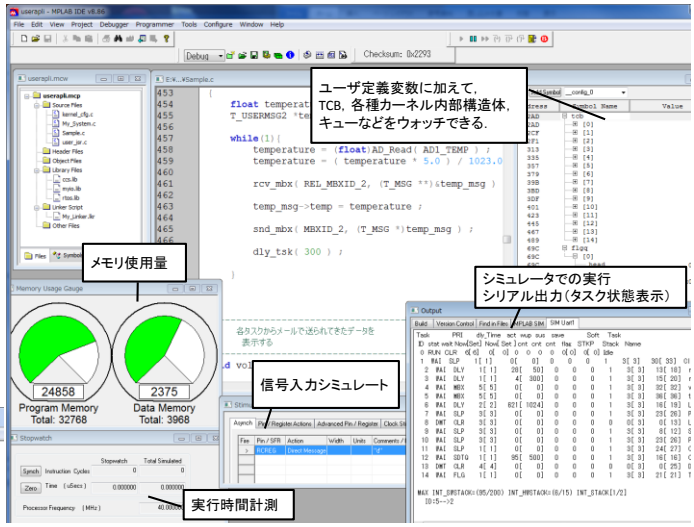

図 7　ユーザプログラム開発画面

６．　授業実践

本学電子情報専攻 2 年前期の「組込みソフトウェア実習（100 分×2 コマ×18 回）」での授業実践例について報告する．「組込みソフトウェア実習」では，リアルタイム OS のサービスコール使用方法とその動作，タスクの優先度・状態遷移・スケジューリングについて理解し，基礎的なマルチタスクプログラミングができることを到達目標としている．

この実習では，各回においてテーマとなるリアルタイム OS 利用における概念及び関連サービスコールの使用法・動作について，30 分程度のスライドによる説明をまず行う．続いて，課題出題，実験実習,レポート提出という形式で実施した．各回の主なテーマは，タスク管理（タスク生成，起動，終了，要求キューイング），タスク付属同期（起床，起床待ち），割込み管理，周期ハンドラ，タスク競合，排他制御，タスク間同期通信（イベントフラグ，データキュー，メールボックス）などである．教材の総スライド数は 148 枚，実習課題は A4 にて 35 ページである．課題としては，各回のテーマに沿って，主に次の 3 パターンを用意した．

(a)シミュレータによる動作解析
　受講生は，シンプルなサンプルコードをシミュレータにて動作解析し，タスクスケジューリングを図示する課題である．優先度や各種パラメータを変更しスケジューリングの変化を確認する．

(b)実験用ボードによる動作解析
　実験用ボードの各種入出力を制御するサンプルコードにおけるタスク動作解析・問題点解析を行う課題である．

(c)プログラム作成
　仕様を満たすプログラムを作成する課題である．

以上の課題構成は，(a)の課題でサービスコールの基本的な動作を理解し，(b)の課題でそれを実際の入出力デバイス制御に適用する方法と注意点を学習し，(c)の課題にて自力でのコード記述・修正が行えることを想定した．

表 2　「組込みソフトウェア実習」アンケート結果

	アンケート内容	平均値
Q1	開発環境は使いやすかったか	2.95
Q2	シミュレータ・デバッガ機能によって，動作解析しやすくなったか	3.23
Q3	タスクの状態遷移を理解できたか	3.09
Q4	タスクスケジューリングを理解できたか	3.05
Q5	説明はわかりやすかったか	3.27

７．　授業アンケート結果

平成 27 年度「組込みソフトウェア実習」の受講生 22 名に対して，授業最終回に 4 件法によるアンケートを行った．設問に対する肯定的評価の順に「そう思う」:4 ポイント，「まあまあ思う」:3 ポイント，「あまり思わない」:2 ポイント，「全然思わない」:1 ポイントとして集計した．対象者は全員，「C 言語プログラミング」及び「マイコンプログラミング（OS なし）」の経験があったが，リアルタイム OS プログラミングについては本実習で初めて経験した．この結果を表 2 に示す．全体平均として 3 ポイント以上となり，概ね肯定的な評価が得られた．また，Q2 と Q3 は相関係数 0.73 となり，やや強い相関関係を示した．TCB などのカーネル内部構造体を確認しながらのデバッグ課題を多数出題しており，これがタスク状態遷移等の理解の手助けとなったと考える

８．まとめ

本稿では，μITRON 仕様準拠 OS の 8 ビット PIC への実装について報告した．本システムでは，PIC の小容量メモリで多数のタスクを並列動作させるために，共有スタック構造を採用した．タスクごとに必要となる最低限のスタック退避領域サイズは，コンフィギュレータにて自動算出できるようにし，利便性を高めた．また，タスク切り替え時間について検証した結果，スタック退避・復帰処理が発生するケースにおいても数十μs 程度でのタスク切り替えが可能であり，実用上問題ないと考える．

本システムは教育用途での使用を主目的として開発を行った．教育現場での使用や自学自習での利便性を考慮し，タスク動作解析のためのログ・デバッグ機能，フリーソフトウェアによる開発環境，PIC ライタ不要な書込み環境，シミュレータのみでも実習可能などの要件を満たす実験・実習環境を構成した．この実習環境を用いて本学電子情報専攻にて授業を行い，概ね良好な評価を得ることができた．

参　考　文　献

[1] 永野宏治,三林光：アイディアを重視した PIC マイコン回路の共同設計演習,工学教育, 60-4, pp.76-81, 2012

[2] 永澤悟,山田朗：中等教育における PIC ボードシミュレータの開発, 日本教育工学会論文誌, 34, pp.145-148, 2010

[3] 篠崎健一,矢鳴虎夫：Web プログラミング技術とマイコン技術を連携した工業高校における情報技術教育の提案, 工学教育, 55-5, pp.33-39, 2007

[4] 千葉悦弥，管隆寿，秋田敏宏，山本美幸：組込み技術を使用したものづくり教育プログラムの実施と評価，工学教育，58-5, pp.18-23, 2010

[5] 経済産業省：2010 年版組込みソフトウェア産業実態調査報告書, 2010

[6] μITRON4.0 仕様書：http://www.t-engine.org/ja/wp-content/themes/wp.vicuna/pdf/specifications/ja/TEF024-S001-04.03.03_ja.pdf　（2016 年 1 月 8 日）

[7] 中谷努，田村仁志，花山英治，小野寺理文，菅野恒雄：電子情報システム工学科における組込み実習教材開発報告，職業能力開発研究発表講演会予稿集，19, pp.132-133, 2011