行政院原子能委員會

委託研究計畫研究報告

核能儀控系統應用正規方法發展與驗證技術研究

Formal Techniques for the Development and Verification of Computerized Nuclear Instrumentation and Control Systems

計畫編號：1022001INER010

受委託機關(構)：國立臺灣大學

計畫主持人：蔡益坤

聯絡電話：(02) 3366-1189

E-mail address：tsay@ntu.edu.tw

核研所聯絡人員：游原昌

報告日期：中華民國 102 年 11 月 27 日

目　錄

中文摘要

本計畫延續前兩個年度在應用正規方法於核能儀控系統認證與驗證方面的研究，主要的方向在於探討如何從即時應用系統的程式碼萃取時間分析所需的工作負載模型，並運用程式邏輯驗證與模型檢測工具，以確認所有工作是否能在預定的時間內完成。基於導入正規方法成效之考量，在本年度的計畫中我們專注於無作業系統的即時監控程式，並以個案分析的方式進行完整的正規分析與驗證的工作。我們以一個在單晶片處理器上執行的電流監控程式為例，說明如何依據處理器中斷與計時機制，從程式碼萃取工作負載模型，進而確認監控程式確實能在電流值過高過久時執行跳脫。

關鍵詞：中斷、模型檢測、Modex、程式證明、即時程式、排程分析、SPIN、時間分析、工作負載模型

Abstract

This project is a continuation of the previous two years' effort in applying formal methods in the certification and verification of computerized nuclear instrumentation and control systems. The main direction has been on investigating how to extract a workload model from the code of a real-time application, and to ensure, with the help of program verification and model checking tools, that all tasks will be completed in time. To maximize the effectiveness of introducing formal methods, we focus in this project on control programs that operate without an operating system kernel, in the hope of obtaining a more complete method for the formal analysis and verification of such programs. We use as an example an electric current monitoring program running on a single-chip microprocessor, to illustrate how to extract a work load model from the code based on the microprocessor's interrupt and timer mechanisms and then ensure that the monitor indeed executes a trip when the current value is too high for too long.

Keywords: Interrupts, Model Checking, Modex, Program Verification, Real-Time Programs, Schedulability Analysis, SPIN, Timing Analysis, Workload Models.

壹、 計畫緣起與目的

從愈來愈多相關的學術研討會活動可以觀察到，應用正規方法（formal methods）[Bowen]於電腦化系統的開發與驗證，已成為軟體工程基礎研究的主要方向之一。這乃由於多年來的實務經驗逐漸顯示正規方法在提高系統品質有其根本且策略的重要性。即使未全程使用正規方法，以嚴謹的數學及邏輯方法與工具驗證軟體程式（即正規驗證）已被視為確保日趨複雜之軟體功能正確無誤的最根本且有效之道[Duval and Cattel 1996][Brat et al. 2004][Hoare 2003]。台灣如欲趕上先進國家自主開發可靠的高安全系統，進而提升整體的電腦軟硬體技術水準，持續投入正規方法相關的研究實有其必要與價值。

本計畫延續前兩個年度在應用正規方法於核能儀控系統認證與驗證方面的研究，主要的方向在於探討如何從即時應用系統的程式碼萃取時間分析所需的工作負載模型，並運用程式邏輯驗證與模型檢測工具，以確認所有工作是否能在預定的時間內完成。前兩個年度的研究著重於在即時作業系統上執行之多執行緒程式的分析與驗證；然而，這樣的分析與驗證必須依賴即時作業系統排程等行為之精確語意模型，相關技術資料取得之困難容易影響正規方法導入與落實的成效。

相對而言，無作業系統的即時應用程式並沒有這樣的限制，較

有利於正規方法之先期導入。核能儀控系統中確實也有較簡單的、無需作業系統便能運作的軟體模組，如感應器監控程式。這類程式之正規驗證無需擔憂複雜之作業系統行為，應是正規方法更能完整發揮的場域。當然，執行程式的處理器硬體仍須以適當語意模型表述其行為，但這比為技術資料不完整的作業系統核心建模要單純許多。

因此，在本年度的計畫中我們以個案分析的方式，專注於無作業系統的即時監控程式，提出針對這類程式較完整的正規分析與驗證的方法。我們以一個在單晶片處理器上執行的電流監控程式為例，說明如何依據處理器中斷與計時機制，從程式碼萃取工作負載模型，進而確認監控程式確實能在電流值過高過久時執行跳脫。

貳、 研究方法與過程

本計畫延續前兩個年度在應用正規方法於核能儀控系統認證與驗證方面的研究。其驗證方法與驗證程序請參考下列兩份報告[28]核能儀控系統電腦化發展之安全功能認證研究，行政院原子能委員會委託研究計畫研究報告，INER- A2510R。及[29]核能儀控系統軟體模組安全功能驗證研究，行政院原子能委員會委託研究計畫研究報告， INER-A2722R。此兩份報告詳述基本靜態程式分析、功能正確性驗證、以及執行時間與排程分析等驗證方法。

因此，在本章中我們將直接說明案例的內容，並解說如何對其功能及時間的正確性進行驗證。

一、範例個案

我們探討的案例是一個在 ADuC841 系列（與 8051 相容）單晶片處理器上執行的電流監控程式。該監控程式以 C 語言，配合針對 ADuC841 相關輸出入埠、暫存器位址及資料型態設定實作，其運作之情境如下圖所示：

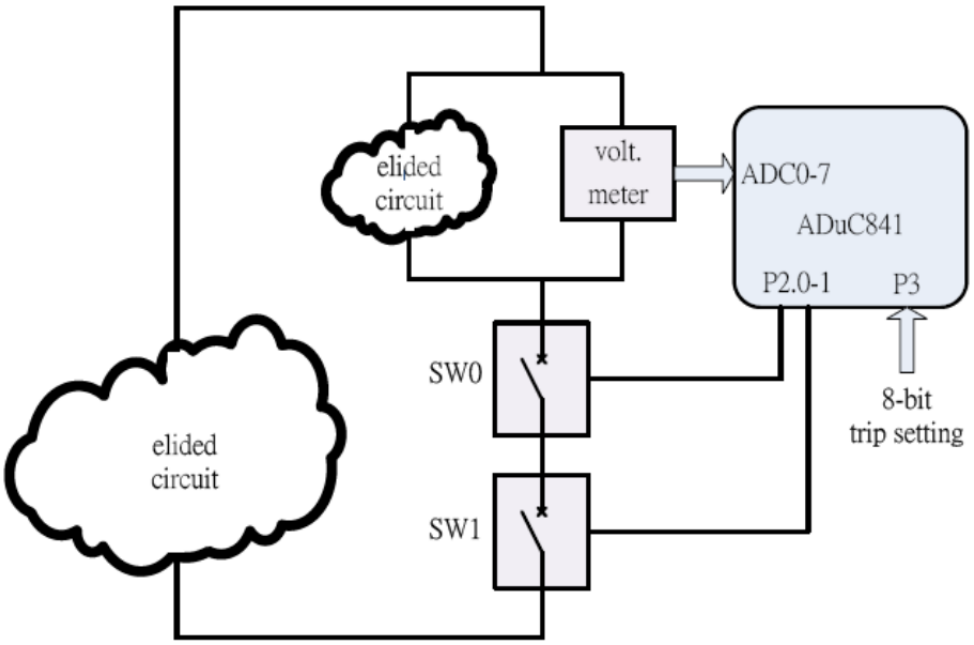

圖 一、電流監控程式運作情境之架構圖

跳脫設定自 P3 輸入埠取得，共 8 個位元；前 4 個位元表示電流值，後 4 個位元示對應之延遲或容忍秒數。其設定對照如下表所示：

表 一、P3 手動設定跳脫設定值

序號	電流值	安培(A)	延遲秒數	秒數(S)
1	0000	0.5	0000	0
2	0001	0.6	0001	1
3	0010	0.7	0010	2
4	0011	0.8	0011	3
5	0100	0.9	0100	4
6	0101	1.0	0101	5
7	0110	1.5	0110	6
8	0111	2.0	0111	7
9	1000	2.5	1000	8
10	1001	3.0	1001	9
11	1010	3.5	1010	10
12	1011	4.0	1011	11
13	1100	4.5	1100	12
14	1101	5.0	1101	13
15	1110	5.5	1110	14
16	1111	6.0	1111	15

SW0 與 SW1 兩個開關的運作方式如下：當 P2.0 輸出位元的值為 0 時，SW0 會打開，使電路成斷路；而當 P2.1 輸出位元的值為 1 時，SW1 會打開，也可使電路成斷路。

電流值之偵測實際上是以ADC輸入埠取得之電壓值換算而成。當偵測到電流值過高時，監控程式會依跳脫設定容忍秒數進入準備跳脫倒數；當高電流持續超過容忍秒數時，則立即執行跳脫，將 P2.0 及 P2.1 兩個輸出位元分別設定為 0 與 1。（電流值正常時，P2.0 及

P2.1 兩個輸出位元的值分別保持為 1 與 0。）

整個電流監控程式分成為計時器中斷處理程序與主程式（含附屬副程式），其基本流程如下圖所示。

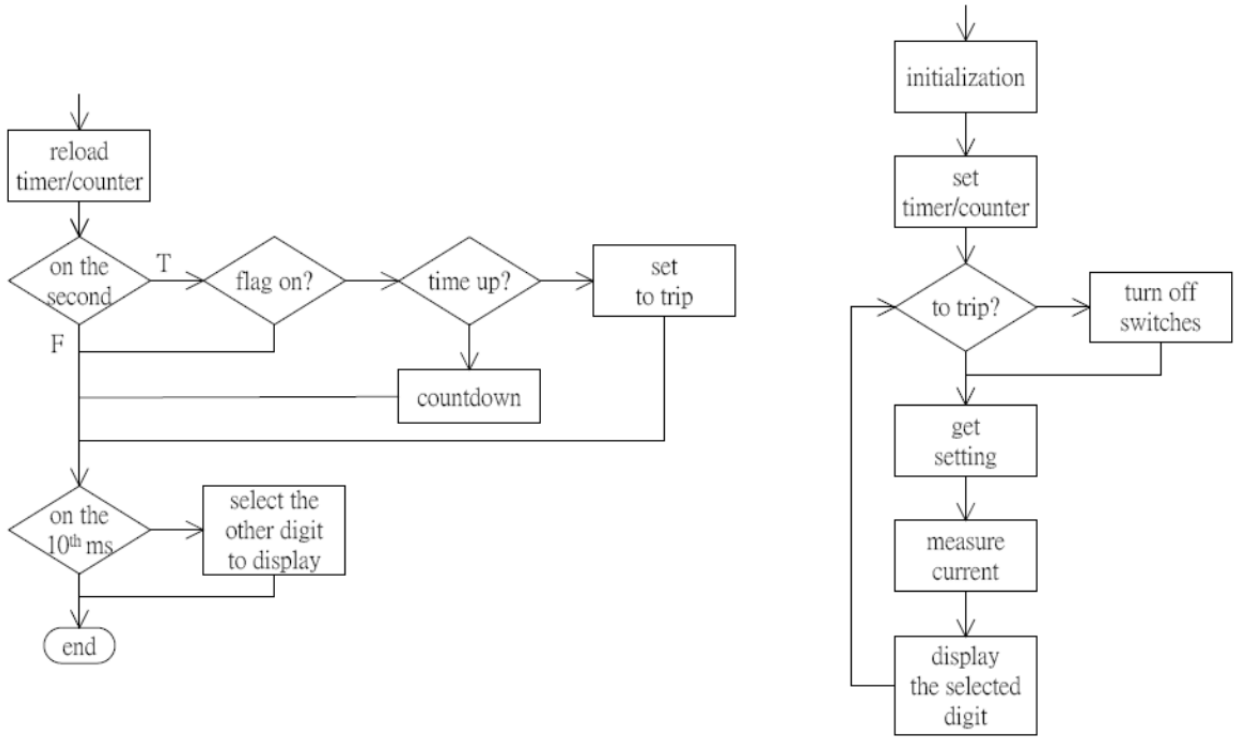

圖 二、計時器中斷處理程序（左）與主程式（右）之基本流程

在主程式量測電流值（measure current）的步驟中，若電流值高於跳脫設定值，則會將一特定號誌（TRIP_FLAG）打開，並設定延遲秒數（TRIP_TIME）之值（註：實際程式碼將時間設定置於 display 的部分），以通知計時器中斷處理程序進行秒數倒數的動作。程式流程中尚包括有顯示倒數秒數的部分，由於此非關鍵功能，在此不說

明其運作之細節。

二、範例個案程式碼

計時器中斷處理程序依石英震盪頻率（目前假設為 11.059 MHz = 11059 KHz），設計為每 1 毫秒執行 1 次。透過計次變數累加，中斷程序每 1000 毫秒即每 1 秒進行電流過高時之秒數倒數；當倒數完畢時，中斷程序打開跳脫指示，伺主程式重掌處理器執行權後（需時遠低於 1 毫秒），便依指示執行跳脫。計時器中斷處理程序之程式碼主體如下。

```
void timer1_handler() interrupt 3
{ // set timer1: interrupt every ms
   TF1 = 0;
   TL1 = 65536 - T; // assuming T KHz (T/1000 MHz) crystal
   TH1 = (65536 - T) >> 8;

   // some other things every 10 ms and every 100 ms; details omitted

   TCA++; // every ms
   if (TCA > 1000) { // every second
      TCA = 0;
      if (TRIP_FLAG) { // high current persists
         if (TRIP_TIME) TRIP_TIME--; // count down
         else {
            TRIP_FLAG = 0;
            TRIP_TIME = 0;
            NORMAL = 0; // to trip
         }
      }
   }
}
```

圖 三、電流監控程式之計時器中斷處理程序

主程式反覆地從輸入埠 P3 讀取跳脫設定，並偵測電流值，電流過高時，啟動倒數計時。若確定跳脫，則改變輸出埠 P2.0 及 P2.1 值，切斷電路。主程式及附屬程序之程式碼主體如下。其中，DIP.BYTE 用來承接跳脫設定 P3 的值，union 資料型態的使用使

DIP.BYTE 這個位元組與 DIP.BIT.TRIP（電流值）及 DIP.BIT.DELAY（延遲時間）所組成的位元組保持等值。

```
union {
   unsigned char BYTE;
   struct {
      unsigned char TRIP:4;
      unsigned char DELAY:4;
   } BIT;
} DIP;

void main(void)
{ // enable timer1 mode; details omitted
   TL1 = 65536 - T; // assuming T KHz crystal
   TH1 = (65536 - T) >> 8;
   // enable interrupts; details omitted
   for (;;)
   { if (NORMAL) {
         PULL_OFF_P21 = 0;
         // some delay by looping; details omitted
         PULL_ON_P20 = 1;
      }
      else {
         PULL_OFF_P21 = 1;
         // some delay by looping; details omitted
         PULL_ON_P20 = 0;
      }
      Read_TRIP_and_Delay_Setting();
      Measure_AC_LOAD_Current();
      Display();
   }
}
```

圖 四、電流監控程式之主程式

```
void Read_TRIP_and_Delay_Setting(void)
{ DIP.BYTE = P3;
   // set TRIP_CURRENT according to DIP.BIT.TRIP; details omitted
}

void Measure_AC_LOAD_Current(void)
{ // get AC_CURRENT; details omitted
   if (AC_CURRENT > TRIP_CURRENT) TRIP_FLAG = 1;
   else TRIP_FLAG = 0;
}

void Display(void)
{ if (TRIP_FLAG) {
      if (!TRIP_TIME_SETTED_FLAG) {
         TRIP_TIME = DIP.BIT.DELAY;
         TRIP_TIME_SETTED_FLAG = 1;
      }
      // display trip time to count down; details omitted
   }
   else {
   // display delay time; details omitted
   }
}
```

圖 五、電流監控程式之附屬副程式

三、功能檢測

無作業系統監控程式的分析與驗證，與需作業系統的監控程式一樣，仍然可分為三大部分：（1）基本靜態程式分析、（2）功能正確性驗證、以及（3）執行時間與排程分析，其中執行時間與排程分析包括工作負載模型的萃取。

我們依據處理器硬體規格及變數資料結構建立監控程式的 SPIN 檢測模型，進而以 SPIN 檢測特定開關變數在中斷程序中被設定後，是否必然觸發主程式應有的後續動作，特別是跳脫的觸發。部分檢測模型的建立亦可使用支援 C 程式語言的 Modex 這套模型萃取工具完成。SPIN 檢測模型建立之主要步驟如下：

一、 依原程式，宣告模型中變數的資料型態，並做必要之近似化。

二、 以獨立行程代表主程式、中斷程序及外部輸入

以下說明這兩個步驟的細節。由於 SPIN 無代表電流值的實數型態，因此近似化以整數替代。由於原監控程式僅處理 16 種可能的電流值門檻，這樣的近似化仍足夠反映監控程式的所有可能行為。此外，就程式功能之邏輯性而言，從測得高電流值到實際觸發跳脫之延遲時間可以縮短，因此與計時配合的計數器 TCA 可以更小的資料型態表示，以提高檢測的速率。最後，DIP 代表 DIP.BYTE，而 DIP_TRIP 及 DIP_DELAY 則代表 DIP.BIT.TRIP 及 DIP.BIT.DELAY，

至於 union 資料型態語意的落實留到模型的可執行部分再完成。

```
byte P3, ADCDATA;
bit NORMAL = 1;
bit PULL_OFF, PULL_ON;
byte TRIP_CURRENT;
byte AC_CURRENT;
bit TRIP_FLAG;
byte TRIP_TIME;
bit TRIP_TIME_SET_FLAG;
byte TCA = 0;
byte DIP;
unsigned DIP_TRIP: 4, DIP_DELAY: 4;
```

圖 六、電流監控程式 SPIN 檢測模型之變數宣告

SPIN 並不支援中斷行為。我們讓中斷處理程式成為獨立的行程（process），使其與其他行程任意交錯出現。以功能而言，這種任意交錯的手法是一種安全的近似做法，也就是說，若檢測結果正確，則原程式也應正確。另外，我們讓計數器 TCA 的上限縮減 10 倍，這有助於驗證速率的提升，卻不致於影響結果的正確性。

```
proctype timer1_handler()
{ do
  :: atomic {
        TCA++;
        if (TCA > 100) { // shortened 10 times
          TCA = 0;
          if (TRIP_FLAG) { // high current persists
            if (TRIP_TIME) TRIP_TIME--; // count down
            else {
              TRIP_FLAG = 0;
              TRIP_TIME = 0;
              NORMAL = 0; // to trip
            }
          }
        };
     };
  od
}
```

圖 七、電流監控程式計時器中斷處理程序之 SPIN 檢測模型

主程式與附屬程式之行為由以下之 main 行程代表。原程式碼以 union 的方式使 DIP 這個位元組與 DIP_TRIP 及 DIP_DELAY 所組成的位元組保持等值，而模型中則以外加設定值的方式落實這樣 union 的語意。

```
proctype main()
{ do
   :: if (NORMAL) {
         PULL_OFF = 0; // delay ignored
         PULL_ON = 1;
      }
      else {
         PULL_OFF = 1;    // delay ignored
         PULL_ON = 0;
      }
      // Read_TRIP_and_Delay_Setting();
      atomic {
         DIP = P3;
         DIP_TRIP = DIP >> 4; // with truncation
         DIP_DELAY = DIP; // with truncation
      };
      TRIP_CURRENT = DIP_TRIP; // simplified
      // Measure_AC_LOAD_Current();
      AC_CURRENT = ADCDATA;
      if (AC_CURRENT > TRIP_CURRENT) TRIP_FLAG = 1;
      else TRIP_FLAG = 0;
      // Display();
      if (TRIP_FLAG) {
         if (!TRIP_TIME_SET_FLAG) {
            TRIP_TIME = DIP_DELAY;
            TRIP_TIME_SET_FLAG = 1;
         };
   od
}
```

圖 八、電流監控程式主程式之 SPIN 檢測模型

有了時間中斷處理程序及主程式的模型後，我們需另加一個行程模擬不同的輸入，包括電流值與跳脫電流值及延遲時間的設定。最後，init 啟動三個行程。

```
proctype env()
{ do
   :: ADCDATA = rand();
        P3 = rand();
   od
}

init {
   atomic {
      run timer1_handler(); run main(); run env();
   }
}
```

圖 九、電流監控程式外部輸入之 SPIN 檢測模型及各行程之啟動

最後是規範並驗證需要檢測的主要性質。第一個性質 p1 確認在從測得高電流值且電流值持續時，監控程式確實會啟動跳脫的步驟。第二個性質 p2 則確認跳脫步驟啟動後，打開電閥的輸出埠之值會正確設定，使電路成為斷路。

```
#define high (AC_CURRENT > TRIP_CURRENT)
#define totrip (NORMAL == 0)
#define open ((PULL_OFF = 1) && (PULL_ON = 0))

ltl p1 { [](high -> (high U (!high || totrip))) }
ltl p2 { [](totrip -> <>open) }
```

圖 十、需要檢測的主要性質之規範

四、時間分析

計時器中斷處理程序各種情境的執行時間可使用 aiT 或量測方式計算。因功能較簡單之單晶片控制器如 ADuC841 系列等無特殊之 cashing 或 pipeline 機制，量測方式也應可以獲致準確的估計值。中斷程序執行一次僅需數十微秒，無法直接量測，因此外加迴圈反覆執行多次，再取平均得到每次執行時間。時間估計並不需要完全精準，只要是很接近實際值的上限即可。

依據 ADuC841 處理器之計時器中斷機制，結合模型檢測及執行時間估算，可為案例監控程式獲致如下的工作負載模型。

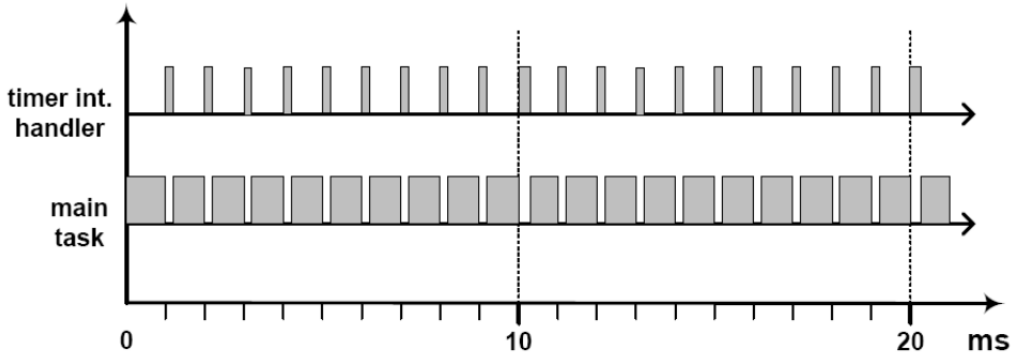

圖 十一、監控程式之工作負載模型

這個負載模型顯示計時器中斷處理程序每 1 毫秒被執行 1 次，其餘時間由主程式掌握處理器執行權，是推論所有工作是否皆如期完成的基礎。下圖顯示測得高電流後，中斷程序在下一個整數秒時開始倒數讀秒，倒數完畢時開啟準備跳脫的開關，接著由主程式執行跳脫的動作。

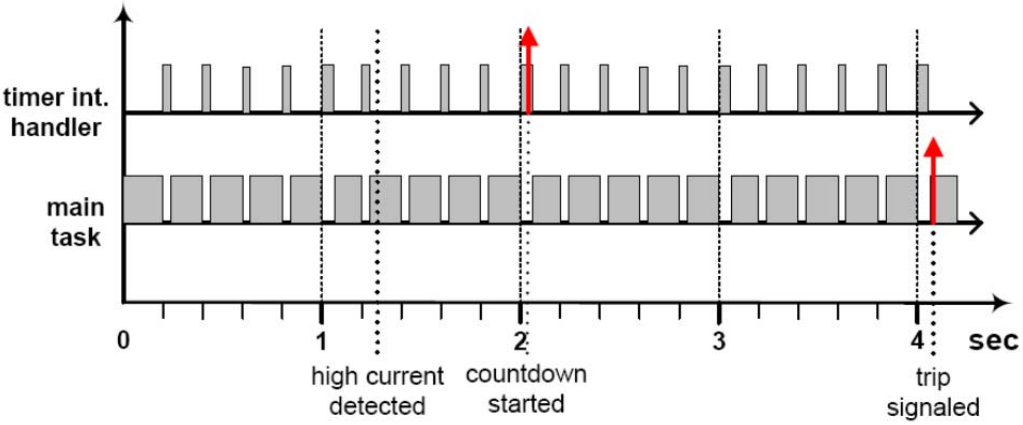

圖 十二、監控程式執行跳脫時之工作負載模型

註：以上工作負載模型的時間軸刻度值較大，無法呈現每 1 毫秒中斷處理程序須完成的工作量。

參、 主要發現與結論

根據我們目前完成的分析與驗證，排除程式碼編譯設定及硬體行為等無法確實檢測之部分，案例監控程式基本上正確無誤，但似乎仍有改善的空間。電流上限及對應容忍時間之設定係於監控系統執行中自外部手動設定，各以 4 個位元表示。該對數值之比例是否合於規格並未於程式中規範，因此無法直接從程式得到驗證。建議之改善方式為：在主程式中，增加一段組態初始化程式碼，明定 16 組與手動設定值一致的電流上限及對應容忍時間值，交付驗證。原跳脫設定之手動輸入仍然保留，但輸入值必須與組態設定比對，確保手動操作合於規範。有了這項變革，監控程式或許可增加自動設定選項，依據組態初始化設定運作。若欲於監控程式執行時改變運作模式，可增設另一開關變數區別自動或手動設定。

此外，自測得高電流值到執行跳脫之實際秒數與設定之容忍秒數最大可能有近一秒左右的差距；改寫計時器中斷處理程式的相關計時部分應可將差距降低至幾毫秒。設定時，將容忍秒數減少一秒是另一種簡單的解決辦法，特別是在程式碼暫時無法修正的情況下。TRIP_TIME 的設定目前於 display 程序中進行，較合理的位置應該是在讀取 P3 值並設定 TRIP_CURRENT 之後立即進行。

依據 MISRA-C 及一般程式撰寫規範，其他缺失尚有：

- 程式撰寫格式

程式碼每行縮排的空白格數不是很一致，基於可讀性原則，應遵循一套標準做法。

- 程序或變數名稱

  名稱若來自於縮寫，可加註解說明。若以英文單字為名稱之一部分，應力求拼字正確，如 delay（原程式碼誤植為 dealy）、set（原程式碼誤植為 setted）。

以上的小缺失，雖不致於影響程式之正確運作，但對於程式的分析驗證及未來的更新維護，多少會造成困擾，值得類似個案借鏡。

對廣義的即時應用程式正規驗證而言，本研究似乎發現了一畝值得耕耘的荒田。計時器中斷處理程序是無作業系統即時應用系統的基礎核心，以適度細微且規律的時間分隔接管處理器執行權，週而復始地檢視各項應定期完成的工作，設定對應的開關變數，再交還處理器執行權，由主程式逐一完成已啟動的工作。很令人意外的，文獻上極少有這類程式正規驗證的探討，完整的理論架構或方法仍付之闕如。本研究之分析驗證方法可進一步嚴謹化、系統化，成為一套驗證即時系統應用程式的架構。

「核能儀控系統」雖有其特殊性，但並不會侷限本研究成果之應用於「核能儀控系統」這個特定領域；事實上各種安全相關電腦化系統之開發與認證皆能受惠於本研究所探討的正規驗證技術。展望未來，本研究之可能延伸發展包括：

- ✓ 應用正規驗證方法與工具於更廣泛之安全相關系統
- ✓ 結合基於軟體模型之測試（model-based testing）與執行中驗證（run-time verification）技術
- ✓ 擴大正規方法之應用於軟體塑模、分析與自動合成
- ✓ 自主正規軟體分析與驗證工具之研發

以上並非全新之研究方向，歐美先進國家早已如火如荼地展開，部分亞洲國家也已跟進。台灣目前在這些方面的研究人口太少、力量有限，但努力耕耘逐步蓄積能量，在安全相關系統的研發與認證上，應該還是可以有所作為。

肆、 參考文獻

2. [MISRA-C: 2004 --- Guidelines for the Use of the C Language in Critical Systems] MISRA-C: 2004 --- Guidelines for the Use of the C Language in Critical Systems. MIRA Limited (2004)
3. [aiT] aiT. Available at: http://www.absint.com/ait/
4. [Bound-T] Bound-T. Available at: http://www.bound-t.com/
5. [Bowen] J. Bowen. The Formal Methods Page. http://formalmethods.wikia.com/wiki/Formal_methods.
6. [Brat et al. 2004] G. Brat, D. Drusinsky, D. Giannakopoulou, A. Goldberg, K. Havelund, M. Lowry, C. Pasareanu, A. Venet, W. Visser, and R. Washington. Experimental evaluation of verification and validation tools on martian rover software. In *Formal Methods in System Design*, vol 25(2-3), pp. 167-198, 2004.
7. [Clarke, Emerson, and Sistla 1986] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. In *ACM Transactions on Programming Languages and Systems*, vol 8, no. 2, pp. 244-263, 1986.
8. [Clarke, Grumberg, and Peled 1999] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*, The MIT Press, 1999.
9. [Duval and Cattel 1996] G. Duval and T. Cattel. Specifying and verifying the steam-boiler problem with SPIN. In *Formal Methods for Industrial Applications*, vol LNCS 1165, pp. 203-217, 1996.
10. [Frama-C software analyzers] Frama-C software analyzers. Available at: http://frama-c.com/
11. [Heckmann and Ferdinand 2004] R. Heckmann and C. Ferdinand. Worst-case execution time prediction by static program analysis. White paper, AbsInt Angewandte Informatik GmbH, 2004. http://www.absint.com/wcet.htm.
12. [Henia et al. 2005] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst. System level performance analysis-the SymTA/S approach. In *IEE Computers and Digital Techniques*, vol 152, no. 2, pp. 148-166, 2005.
13. [Hoare 2003] C.A.R. Hoare. The verifying compiler: A grand challenge for

computer research. In *Journal of the ACM*, vol 50, no. 1, pp. 63-69, 2003.

14. [Holzmann 2003] G. J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*, Addison-Wesley, 2003.
15. [Künzli et al. 2007] S. Künzli, A. Hamann, R. Ernst, and L. Thiele. Combined approach to system level performance analysis of embedded systems. In Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis, pp.63-68, Salzburg, Austria, 2007.
16. [Lamport 2005] Leslie Lamport 2005, 'Real Time is Really Simple', *Technical Report MSR-TR-2005-30*, 2005.
17. [Lehoczky, Sha, and Ding 1989] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm exact characterization and. In Proceedings of Real Time Systems Symposium 1989, pp.166-171, Santa Monica, CA, USA, 1989.
18. [Liu and James 1973] C. L. Liu and W. Layland James 1973, 'Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment', *Journal of the ACM*, 1973, pp. 46-61.
19. [Martin and Leslie] Abadi Martin and Lamport Leslie, 'An Old-Fashioned Recipe for Real Time', *Research Report 91, Digital Equipment Corp., Systems Research Center, Palo*.
20. [Modex] Modex. Available at: http://cm.bell-labs.com/cm/cs/what/modex/
21. [Muhammad and Nahida 2011] I. H. Muhammad and S. C. Nahida. A Practical approach on Model checking with Modex and Spin. In *International Journal of Electrical & Computer Sciences IJECS-IJENS*, no. 5, pp. 1-7, 2011.
22. [RapiTime] RapiTime. Available at: http://www.rapitasystems.com/rapitime
23. [RT-Druid] RT-Druid. Available at: http://www.evidence.eu.com/content/view/28/51/
24. [SPIN] SPIN. Available at: http://spinroot.com/spin/whatispin.html
25. [SWEET] SWEET. Available at: http://www.mrtc.mdh.se/projects/wcet/sweet.html

26. [SymTA/S] SymTA/S. Available at: http://www.symtavision.com/symtas.html
27. [Wilhelm et al. 2008] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström. The worst-case execution-time problem - overview of methods and survey of tools. In *ACM Transactions in Embedded Computing Systems*, vol 7, no. 3, pp. 36-1, May 2008.
28. 核能儀控系統電腦化發展之安全功能認證研究，行政院原子能委員會委託研究計畫研究報告，100 年，INER- A2510R。
29. 核能儀控系統軟體模組安全功能驗證研究，行政院原子能委員會委託研究計畫研究報告，101 年，INER-A2722R。