ARM

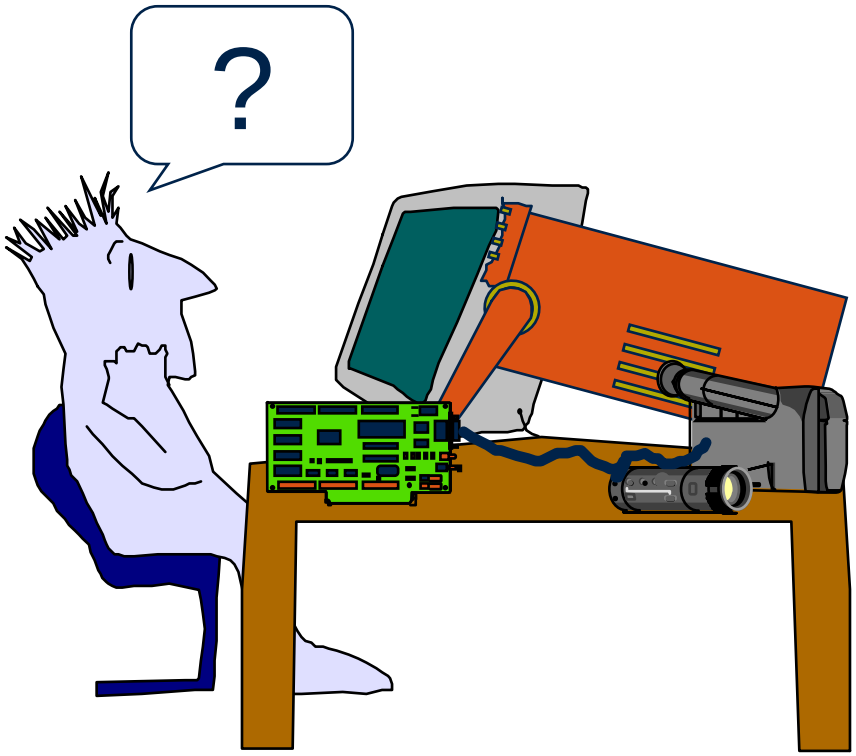
ARM 調試方案

議程

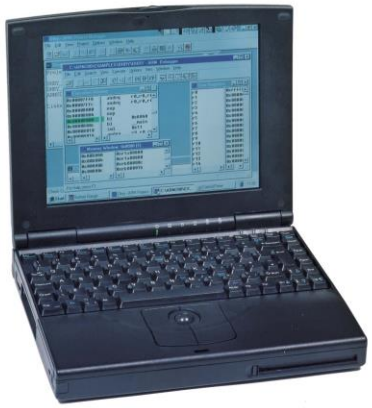
- 基本的調試需求
  - 你需要什麼樣的功能?
  - ARM公司的調試和開發工具。
- 嵌入式核的調試
  - 實現和利用JTAG的調試方案
  - 停止模式和監控模式
- 嵌入式跟蹤
  - 使用ETM
- ARM 開發板

基本的調試需求

- **運行控制**
  - 設置資料訪問中斷點
  - 設置指令中斷點
  - 代碼的單步執行
- **狀態控制**
  - 處理器狀態
    - 讀寫寄存器值
  - 系統狀態
    - 系統記憶體訪問
    - 下載代碼
- **執行歷史**
  - 執行跟蹤資訊
  - 記憶體訪問歷史



ARM 調試系統元件



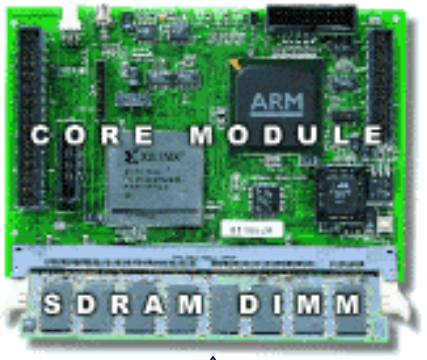
執行AXD調試器的主機
(ADS的一部分)
和 Multi-ICE server

並口

協定轉換器
(Multi-ICE)

JTAG Port

開發板
(Integrator Core Module)



Trace Port

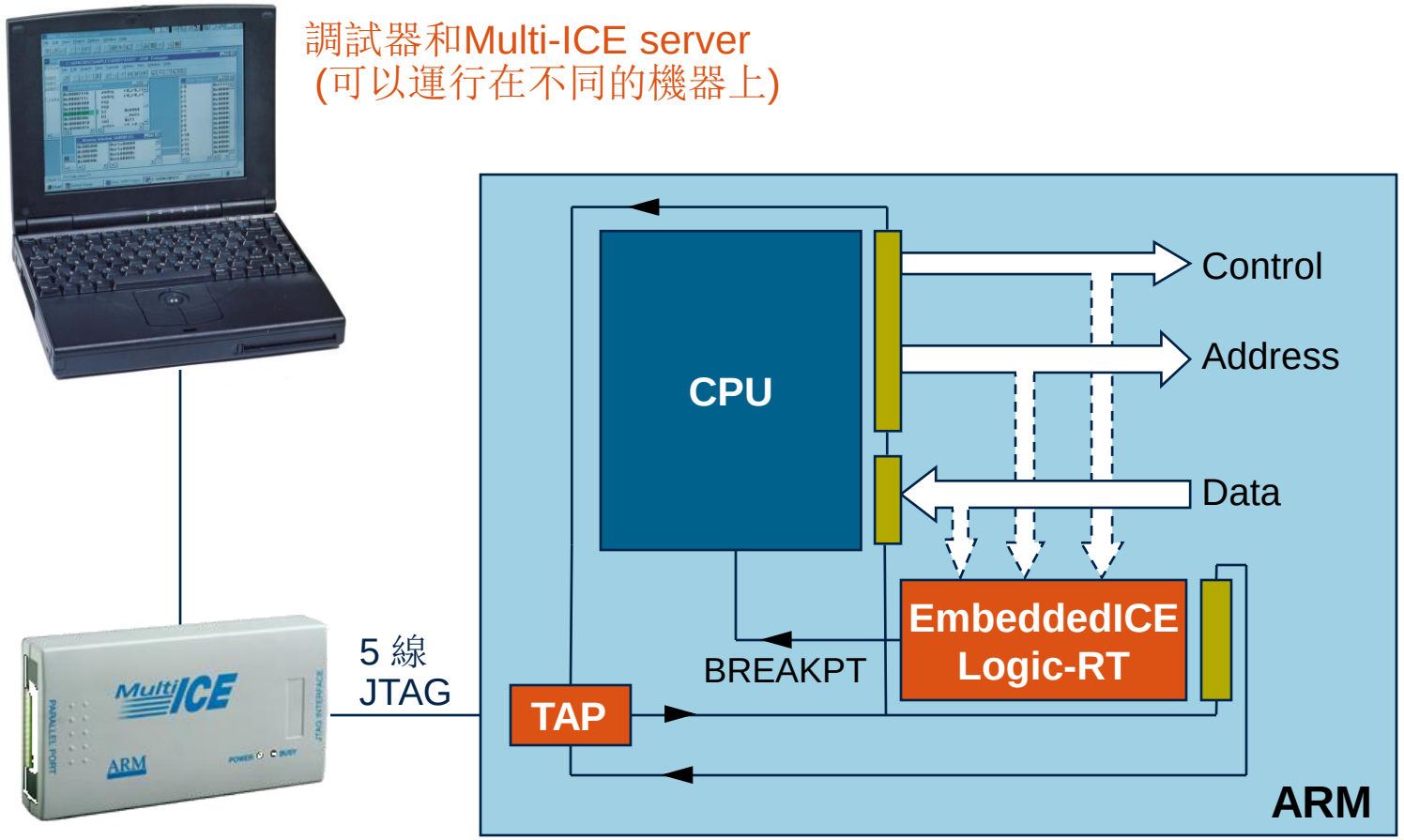
串口I/乙太網

跟蹤介面單元
(MultiTrace)

議程

- **基本的調試需求**
  - 你需要什麼樣的功能?
  - ARM公司的調試和開發組成工具。
- **嵌入式核調試**
  - 實現和利用JTAG的調試方案
  - 停止模式和監控模式
- **嵌入式跟蹤**
  - 整體化和利用ETM
- **ARM 開發板**

嵌入式核調試



- 被調試的系統可以是最終的系統！
- 也可以用協力廠商的協定轉換工具：
  - http://www.arm.com/DevSupp/ICE_Analyz/

嵌入式ICE邏輯

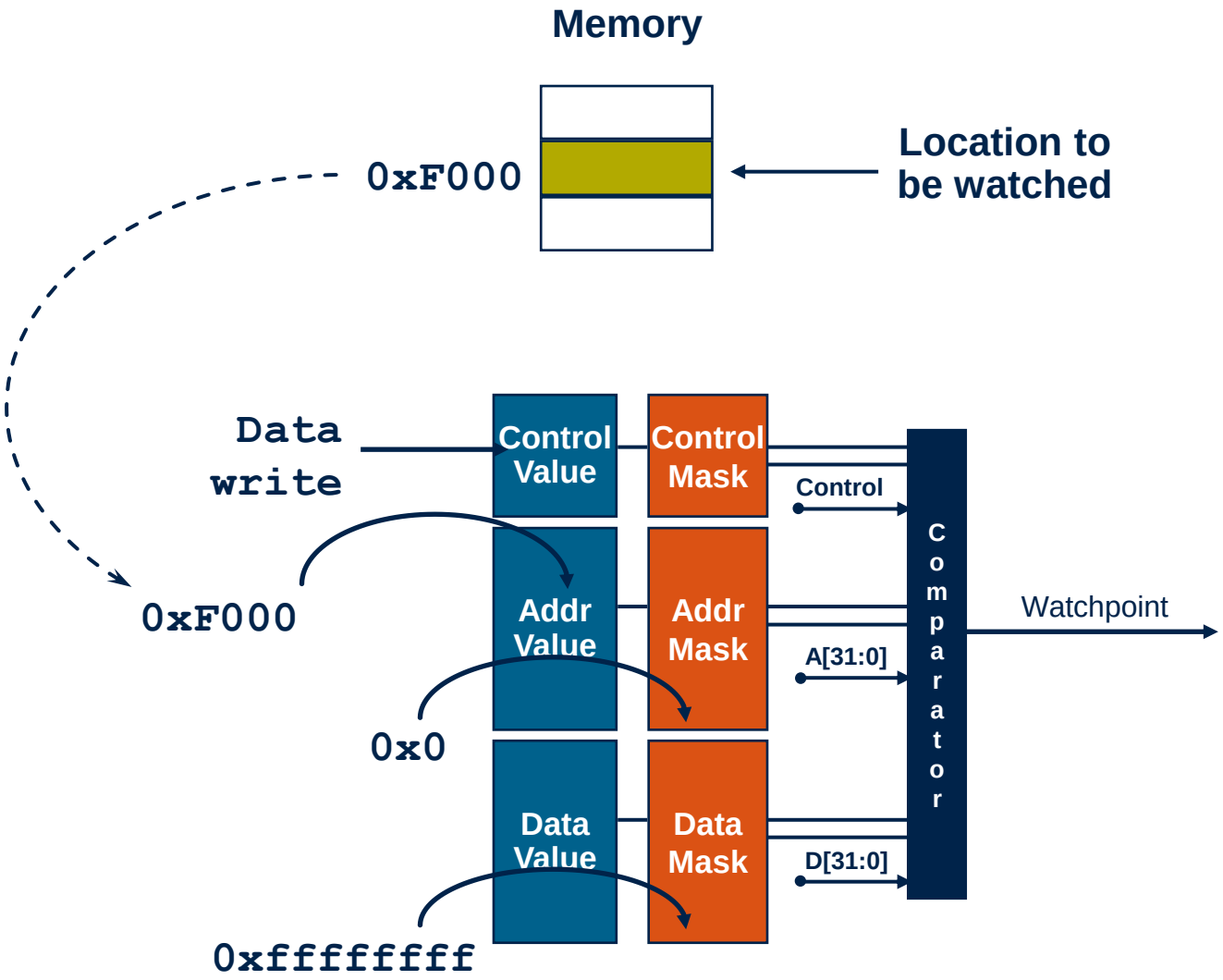
- **兩個觀察點單元**
  - 可以通過監控位址匯流排，資料匯流排和控制信號來探測觀察點（watchpoint）和中斷點。
  - 每個單元可以用來提供
    - 1 觀察點, 或
    - 1個 ROM或RAM裡的硬體中斷點，或
    - RAM裡的多個軟體中斷點
- **調試控制和狀態寄存器**
- **調試通訊通道**
- **注意：ARM10**
  - ARM10家族的調試結構是不同的，雖然原理是一樣的。
  - 一共包括8個觀察點單元
    - 6 個在指令位址匯流排上
    - 2個在資料位址匯流排上
  - 這個將在另外一個課題中講解

觀察點

一個**觀察點**就是一個中斷點，這個中斷點在當以某種方式訪問特定記憶體區域時被觸發.

這個例子裡，當向位址0xF000 寫時，將觸發這個觀察點.

每一個觀察點單元可以設置成一個**觀察點**，而且只能設置一個.



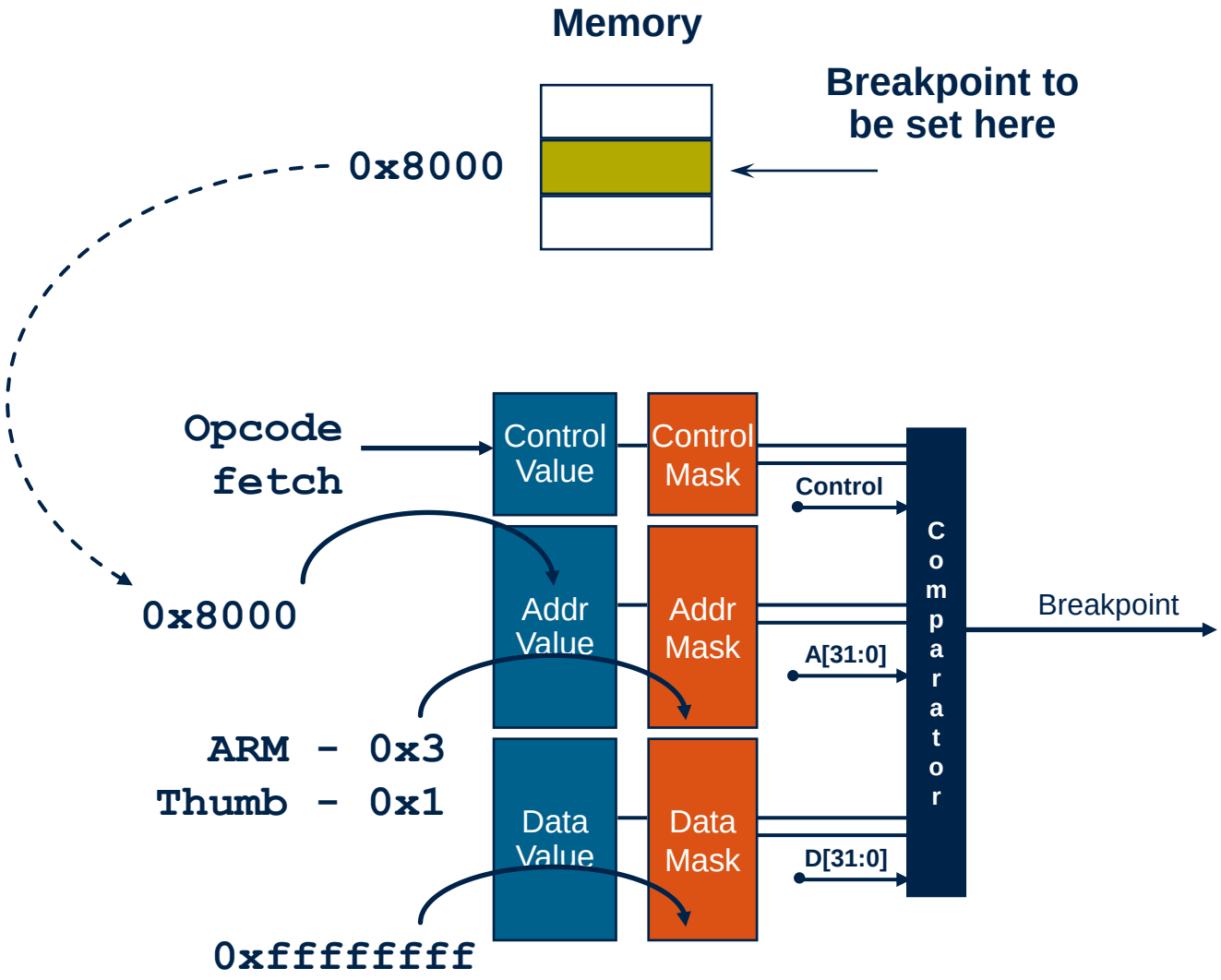
硬體中斷點

當內核試圖執行記憶體裡一個特定位址的指令時，將會觸發一個**硬件中斷點**.

這個例子裡，當要執行從位址0x8000索取的指令時，將會觸發一個**硬件中斷點**.

硬體中斷點可以在RAM或ROM裡設置.

每個觀察點單元可以被用來設置一個**硬件中斷點**，而且只能一個.



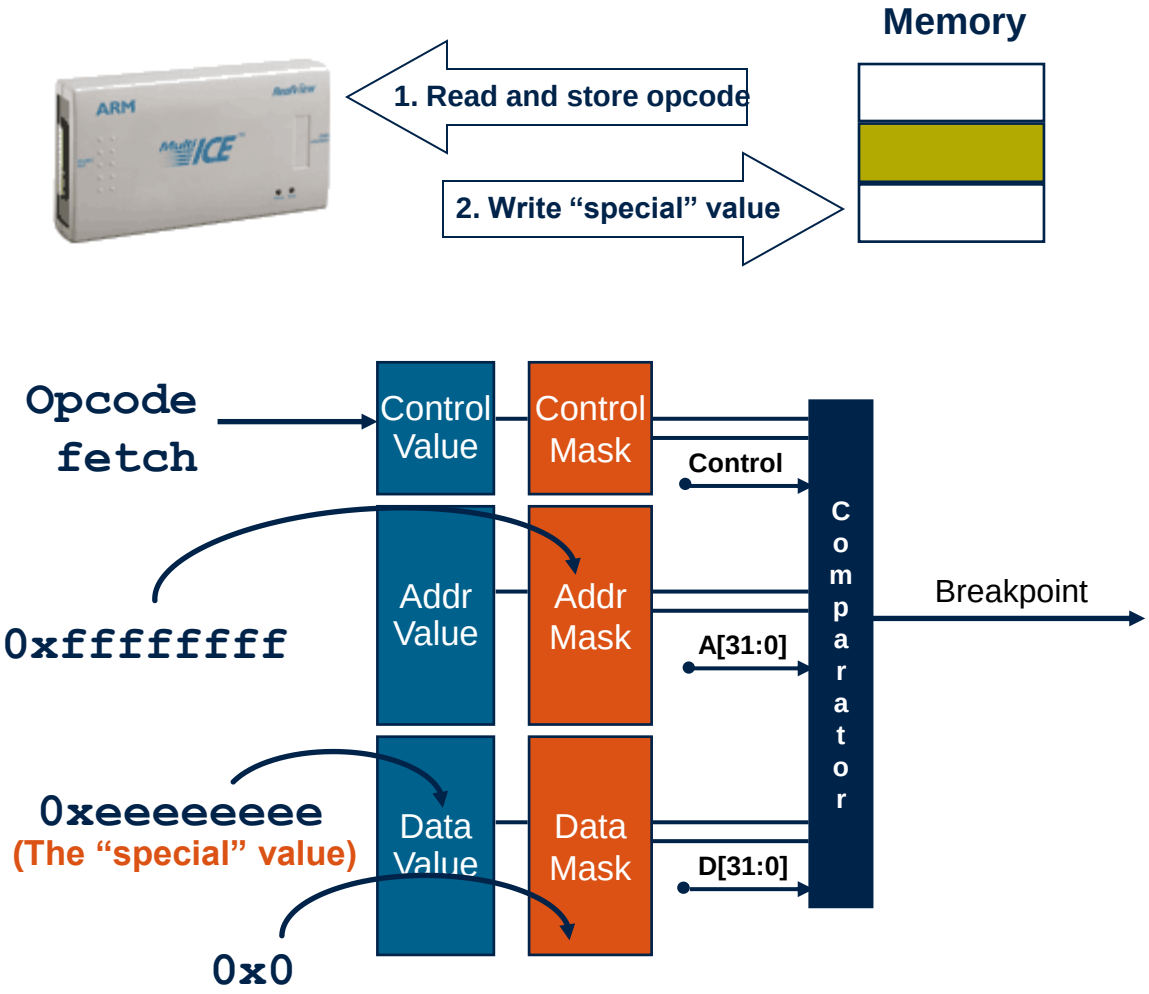
軟體中斷點

一個**軟件中斷點**是當一個特定的指令從任何位址被預取時觸發的一個中斷點.

這個例子表明了觀察點單元的配置 – 這對所有的軟體中斷點都是一樣的.

為了設置一個中斷點，可以使用Multi-ICE在特定的地方寫一個特別的指令．這些只能在RAM裡在操作.

每一個觀察點單元可以用來設置無數的**軟件中斷點**.



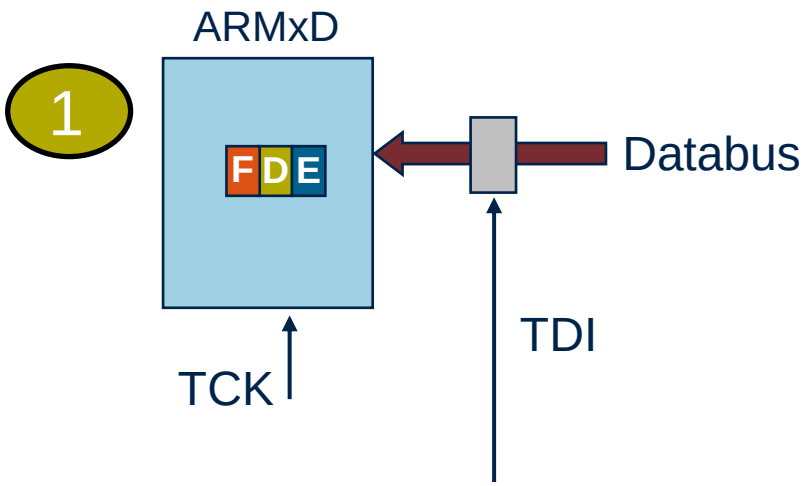
命中一個中斷點

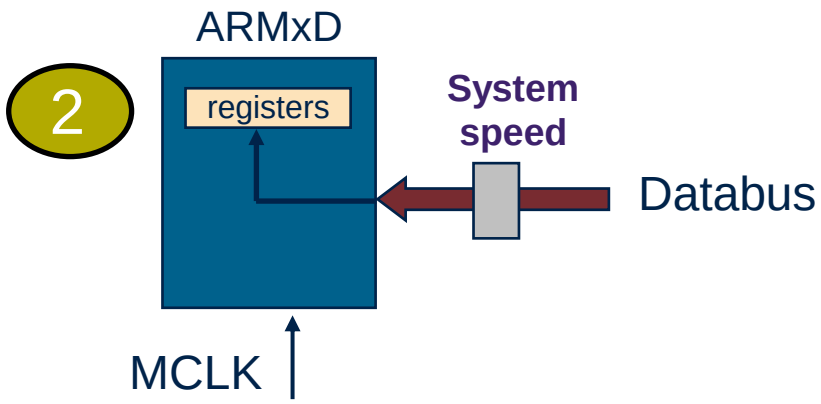
- **停止模式調試**
  - 由標準的 EmbeddedICE & EmbeddedICE-RT支持
    - 內核進入調試狀態和停止狀態
    - 內核與系統的其餘部分分離
    - 發出DBGACK信號 (調試器通過JTAG而檢測到)
    - 沒有中斷處理，除非調試器重新啟動執行代碼
    - 處理器和系統的狀態可以通過掃描內核的指令來察看和修改，緊接著執行它們.

- **監控模式調試**
  - 只由增強功能的 EmbeddedICE-RT支持
    - 在ARM9E, ARM10 和稍後的ARM7TDMI版本上有
    - 內核通過一個異常中斷而進入常駐軟體監控程序
    - 可以連續處理中斷
    - 處理器和系統的狀態可以通過監控程式的調試命令來察看和修改.

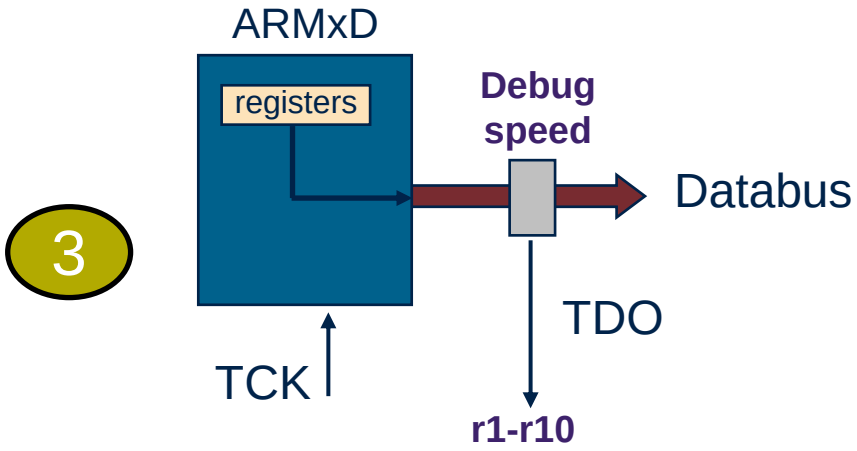
讀系統狀態 (在停止模式)

從地址0x8000 向後讀 10 個字





MOV r0, #0x8000
LDMIA r0!, {r1-r10} - system speed
STMIA r0, {r1-r10} - debug speed



監控模式調試

- 主機 - 在AXD 和 Multi-ICE之間的控制器
- 實時監控協議
  - 非常簡單的協議
  - 快速 – 沒有出錯檢查
  - 允許後臺命令
- 實際目標板
  - 小的調試監控程序 (<2k 代碼位元組)
  - 集成在目標裡.
  - 以目標代碼和原代碼的形式提供

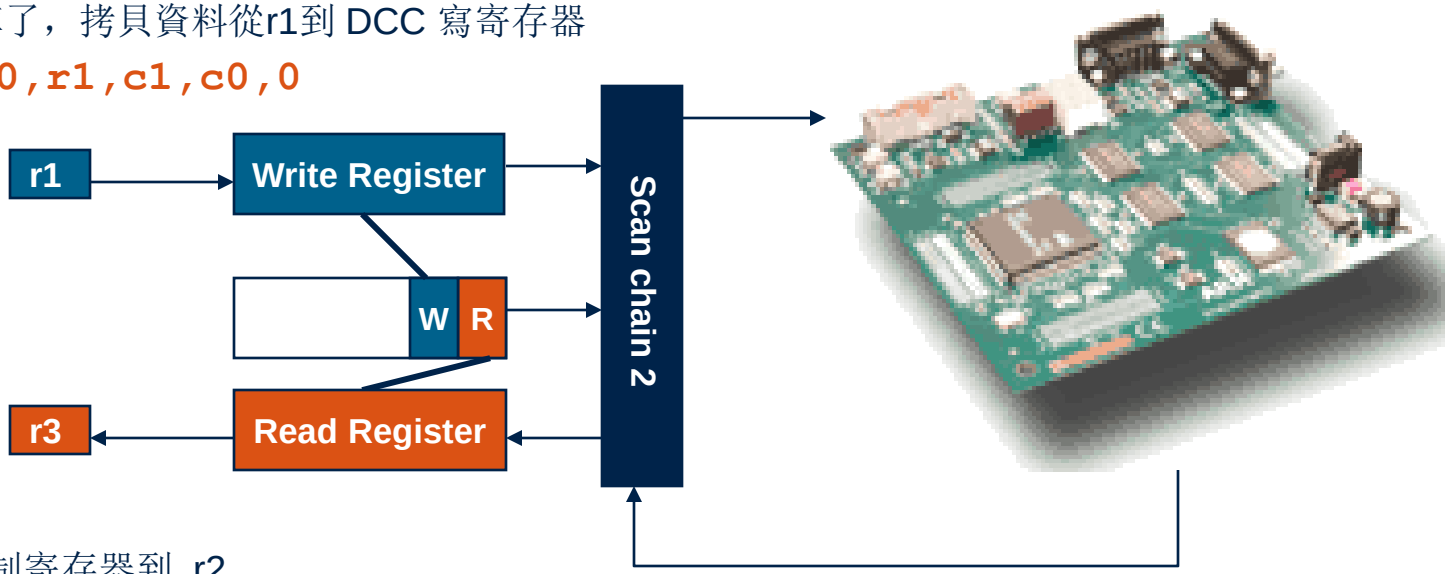
調試通訊通道

- 在ARM和主機調試器之間通過JTAG的通訊資訊是由簡單串列通訊口來實現的.
  - 不進入調試模式和停止程式執行.
- ARM上運行的應用代碼經過輔助處理器14來訪問.
- 三個寄存器
  - 通訊資料讀寄存器（Comms Data Read Register）
    - 主機到ARM的通訊
  - 通訊資料寫寄存器（ Comms Data Write Register）
    - ARM到主機的通訊
  - 通訊資料控制寄存器（Comms Data Control Register）
    - 提供在ARM和主機之間同步的握手信號
    - 位 1 – 寫位 - 當ARM寫入新資料時清掉
    - 位 0 – 讀位 – 當ARM有新數據讀時設置

DCC 查詢模式通訊

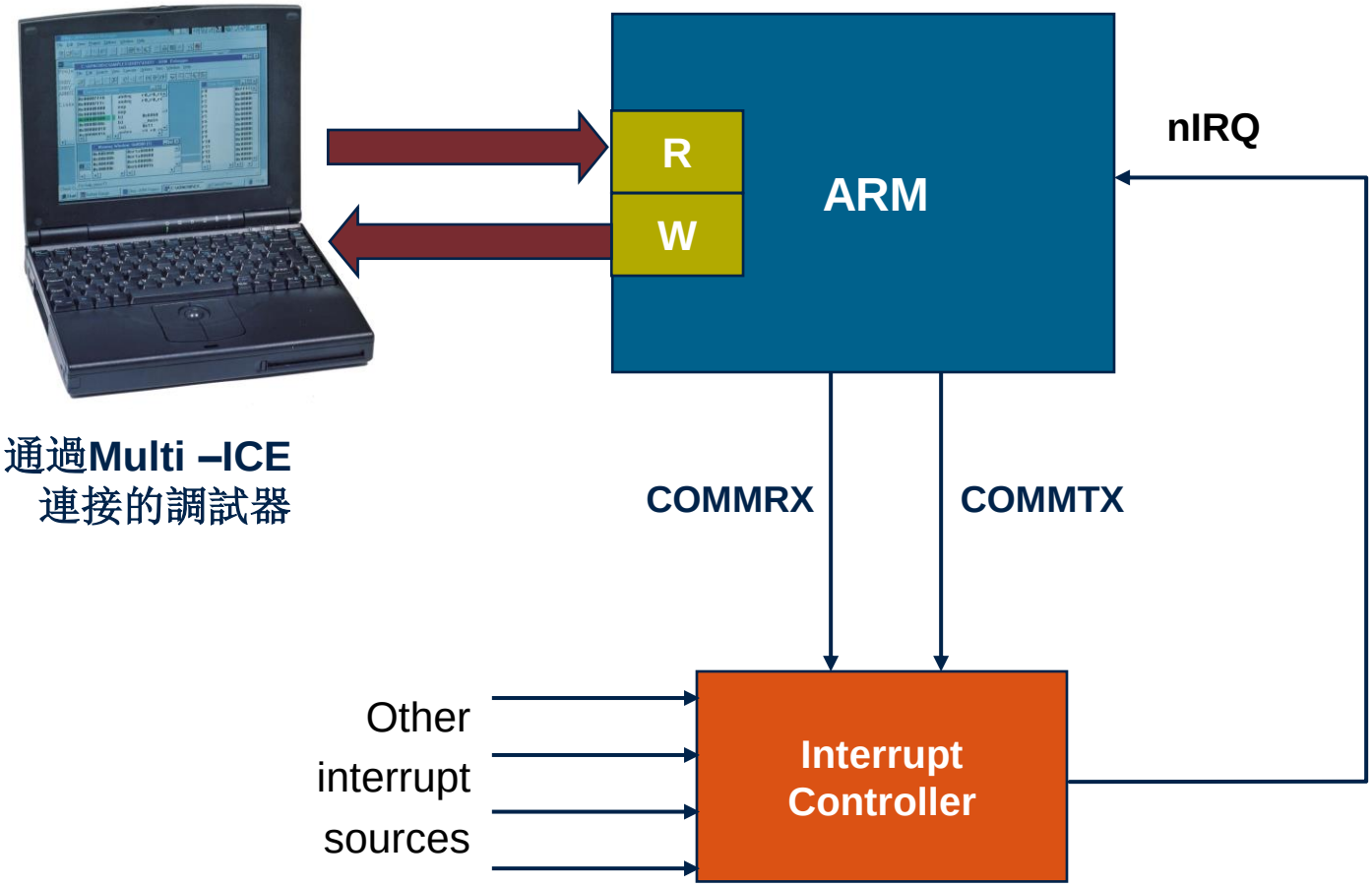
```
; 拷貝 DCC 控制寄存器到 r2
MRC p14,0,r2,c0,c0,0
; 檢查DCC 控制寄存器的位1
TST r2,#0x2
; 如果位元1清掉了，拷貝資料從r1到 DCC 寫寄存器
MCREQ p14,0,r1,c1,c0,0
```

調試器查詢控制寄存器利用掃描鏈2來察看什麼時候寫位被設置，讀位被清掉．接著資料可以被掃描進入或者輸出．



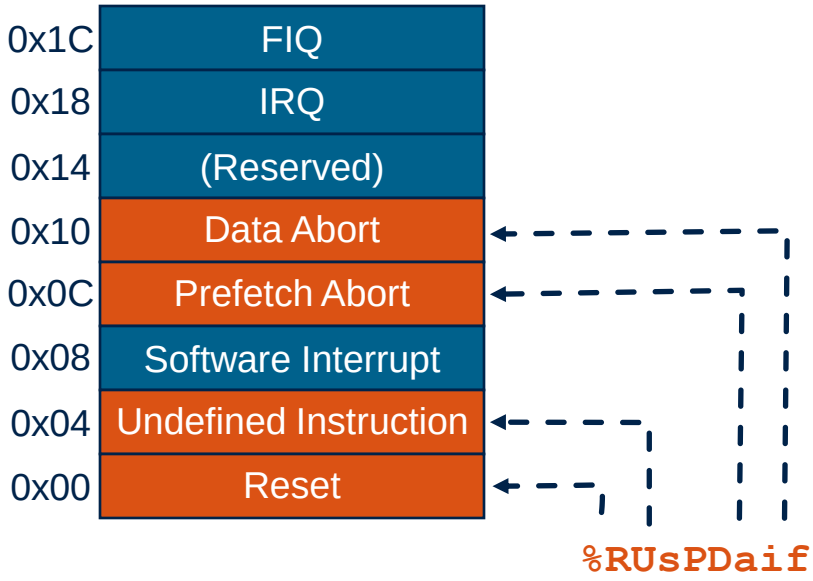

```
; 拷貝 DCC 控制寄存器到 r2
MRC  p14,0,r2,c0,c0,0
;檢查DCC 控制寄存器的位0
TST r2,#0x1
; 如果位元0設置，拷貝資料從DCC 寫寄存器 到r3
MRCNE  p14,0,r3,c1,c0,0
```

DCC 利用中斷進行通訊

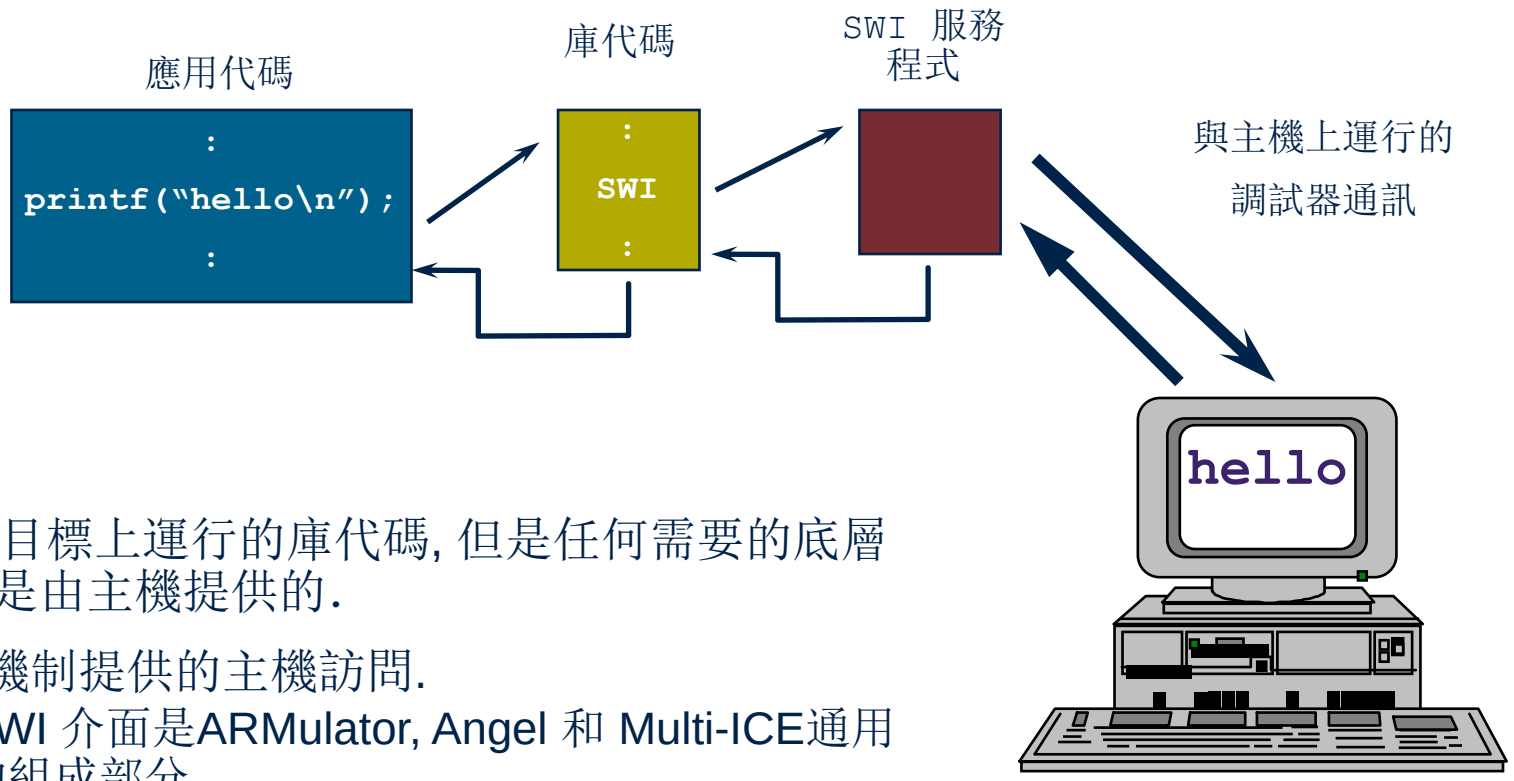


向量捕獲

- 允許沒有處理軟體的異常機制的捕獲
- 在ARM7TDMI 上執行利用中斷點
  - 當從ROM 地址 0x0 調試時關閉
- 在ARM9TDMI/ARM10和後來的版本上執行利用專門的硬體
  - 只對硬體異常敏感.
  - 到向量表裡的跳轉將不被捕獲
  - 留下觀察點單元作為一般使用.
- 一旦有你自己的處理，則關掉向量捕獲.
  - AXD : `spp vector_catch 0`



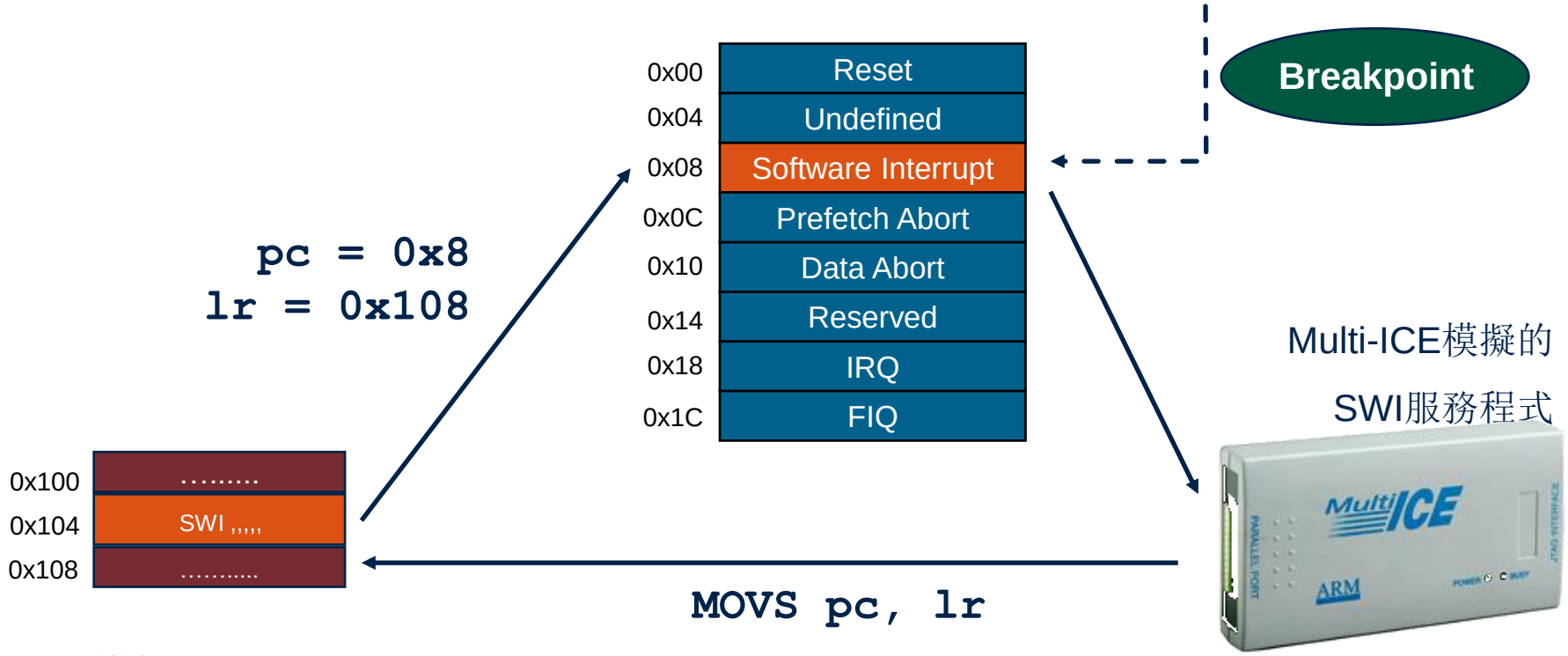
Semihosting



- ARM 目標上運行的庫代碼, 但是任何需要的底層的I/O是由主機提供的.
- SWI 機制提供的主機訪問.
  - SWI 介面是ARMulator, Angel 和 Multi-ICE通用的組成部分.
  - Semihosted 程式將運行在所有的ARM目標板上，而不需要 移植.
- 需要連接的調試工具提供這些功能.

通過JTAG 的Semihosting

- spp semihosting_vector 0x8
- spp semihosting_enabled 1

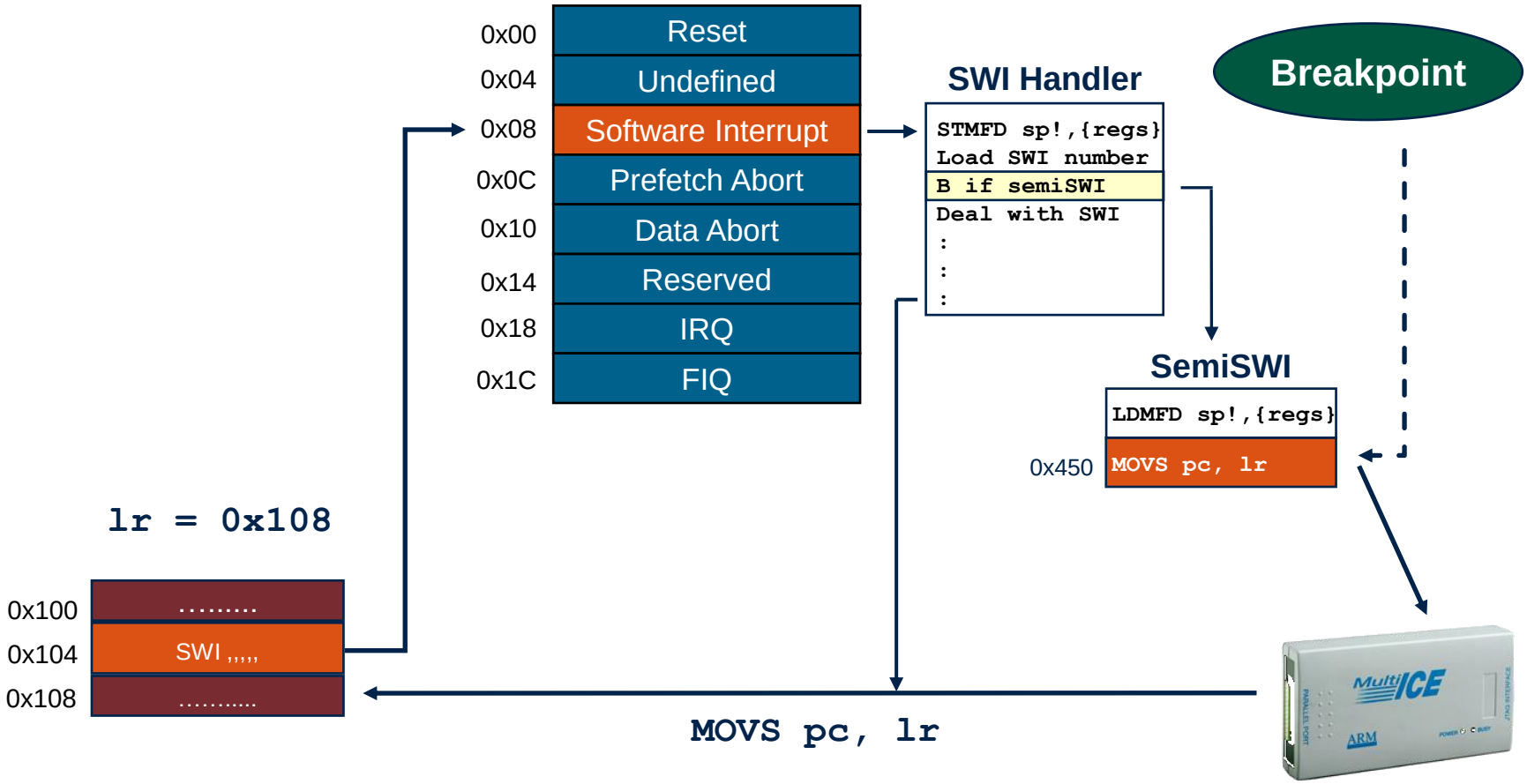


- 關掉:

```
spp semihosting_enabled 0
```

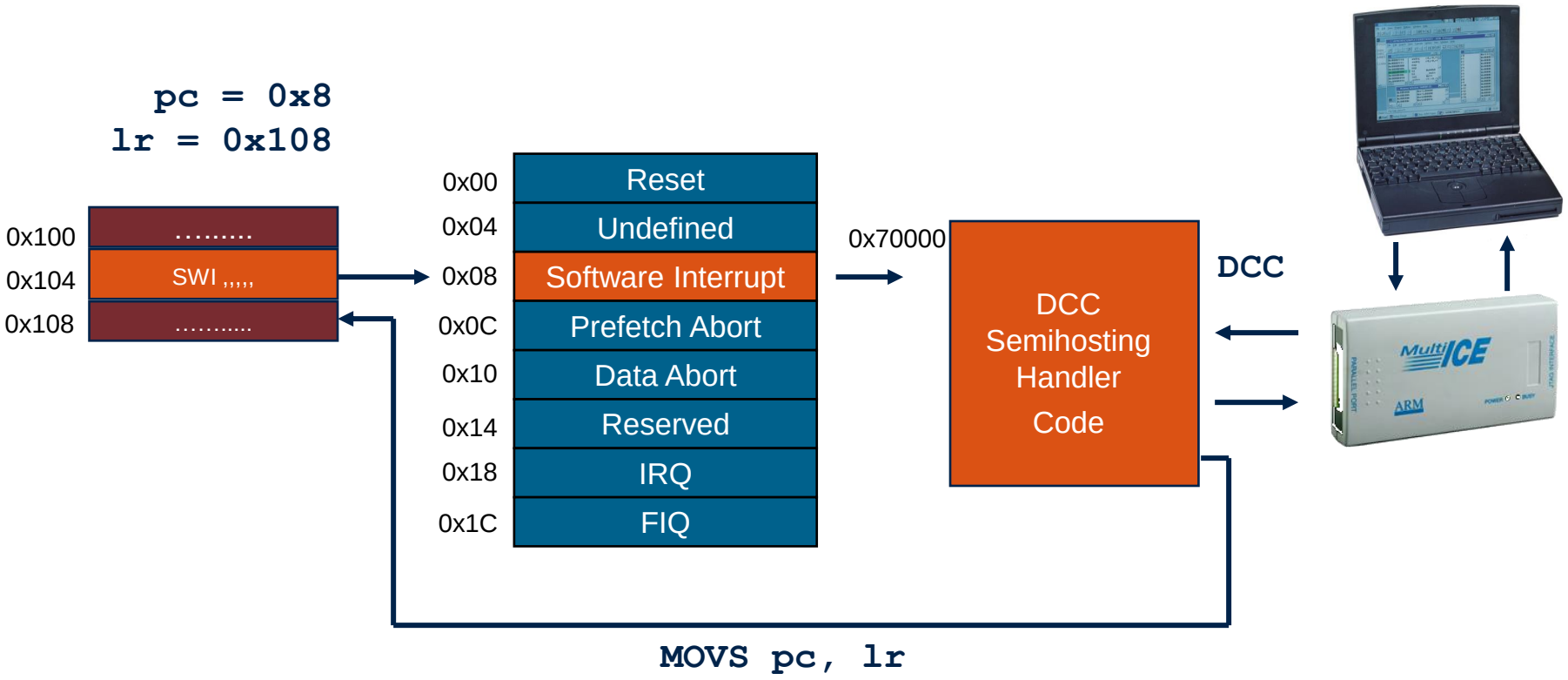
Semihosting 加上用戶SWIs

- `spp semihosting_enabled 1`
- `spp semihosting_vector 0x450`



DCC Semihosting

- `spp semihosting_dcchandler_address 0x70000`
- `spp semihosting_vector 0x8`
- `spp semihosting_enabled 2`



直接對EmbeddedICE訪問

- AXD 允許直接對EmbeddedICE 邏輯寄存器訪問
- 可以直接通過 GUI或如下的命令列察看

```
REGisters "EICE Watch 0"
```

在地址0x4000寫時設置一個觀察點:

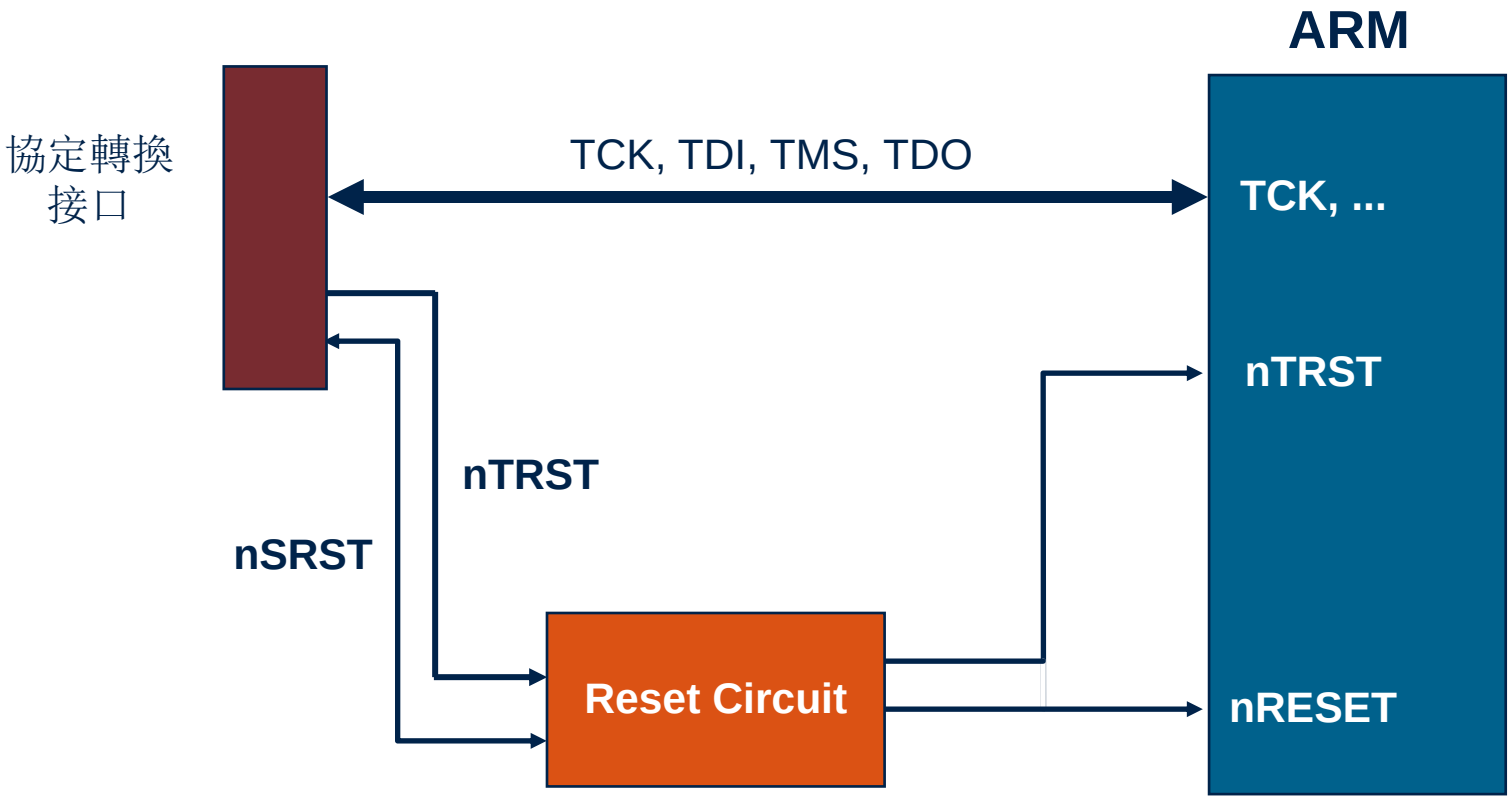
```
sreg "EICE Watch 0|Address Value" 0x00004000
sreg "EICE Watch 0|Address Mask" 0
sreg "EICE Watch 0|Data Value" 0
sreg "EICE Watch 0|Data Mask" 0xFFFFFFFF
sreg "EICE Watch 0|Control Value" 0x10D
sreg "EICE Watch 0|Control Mask" 0x0F8
```

- 注釋: ARM調試器優先與 ADS1.1 通過輔助處理器0訪問EmbeddedICE 邏輯寄存器.

利用Multi-ICE調試設計

- **確信 DBGEN 是接高電平!**
- **Thumb 代碼上的軟體中斷點需要半字訪問RAM**
  - 必須總是字/半字/位元組訪問記憶體
- **Multi-ICE可以在1.0V 到 5.0V時調試系統**
  - 自適應到目標邏輯電壓
- **目標板可以在>2.0V時工作**
  - 在3.3V時，啟動電流是~400mA， 一般操作時是 ~120mA
  - Multi-ICE 2.1 出售時跟隨一個適當的電源
- **Multi-ICE用20-針 連接器**
  - 多個接地點
  - 靈活的時鐘配置
    - 可以與不同的時鐘速度設備同步
    - 長 JTAG 電纜
  - 也可以僅僅利用5 個信號來調試
- **請小心停止系統時鐘**
- **復位考慮…..**

復位



- nTRST 和 nSRST 必須通過一個上拉電阻來連接
- Multi-ICE 在 nTRST 有開放的連接

測驗

1) EmbeddedICE 邏輯一共包括多少個觀察點單元?

2) ROM上可以設置多少個中斷點?

3) 利用DCC semihosting比一般的 semihosting有什麼優勢?

4) 為了啟動ARM的調試功能，DBGEN 的信號應該怎麼連接?

5) 為什麼Multi-ICE需要半字訪問記憶體?

6) EmbeddedICE-RT 提供了哪些額外的調試功能?

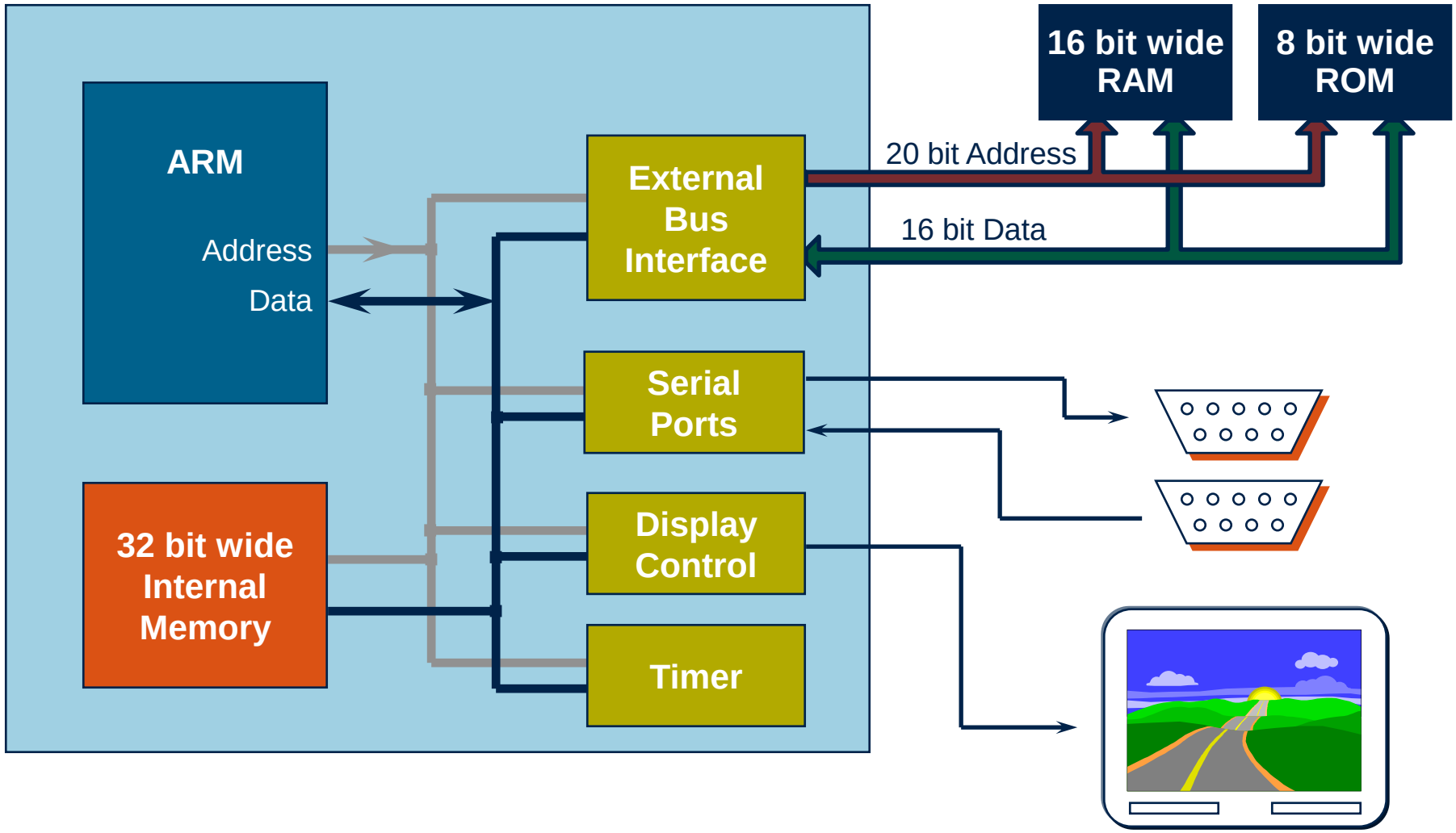
議程

- **基本調試需求**
  - 你需要什麼樣的功能?
  - ARM公司的調試和開發集成工具。
- **嵌入式核調試**
  - 實現和利用JTAG的調試方案
  - 停止模式和監控模式
- **嵌入式跟蹤**
  - 整體化和利用ETM
- **ARM 開發板**

ETM的機制
(嵌入式跟蹤巨集單元)

- **為什麼需要即時跟蹤功能?**
  - 即時系統不允許停止!
    - 傳統的調試 (中斷點和單步執行) 不能滿足
    - 必須利用捕獲的實際代碼運行來調試
- **為什麼不用其它的調試工具而用ETM?**
  - EmbeddedICE-RT為調試通訊通道提供了低的頻寬
    - 只適用於程式狀態資訊
    - 外部的指令需要寫到通訊通道裡
- **獨立的處理器比嵌入式處理器容易調試**
  - 一個獨立的處理器，或者:
    - 用外部的ICE單元代替處理器，或者
    - 用邏輯分析儀探測處理器的信號
  - 當用嵌入式ARM內核時兩個都不可能時怎麼辦啊?

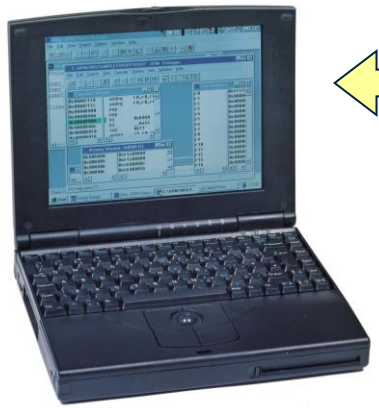
帶嵌入式處理器的典型ASIC



問題

- **沒有外部可見的片上ASIC 匯流排**
  - 帶緩存的處理器 (例如, ARM920T) 帶有與設備內部相連的核/緩存匯流排
- **ICE 版本必須在全速系統速度下運行**
  - 隨著處理器速度的增加，ICE越來越難.
  - 同時必須提供ICE 工具 (例如，觸發器)
- **即時跟蹤需要確定的高頻寬的介面**
  - 需要位址匯流排，資料匯流排和控制信號
  - 對於 ARM7TDMI ，有80多個 引腳
- **很多 ASICs 使用相同的處理器核**
  - ICE必須為每一個 ASIC定做

ARM 即時跟蹤

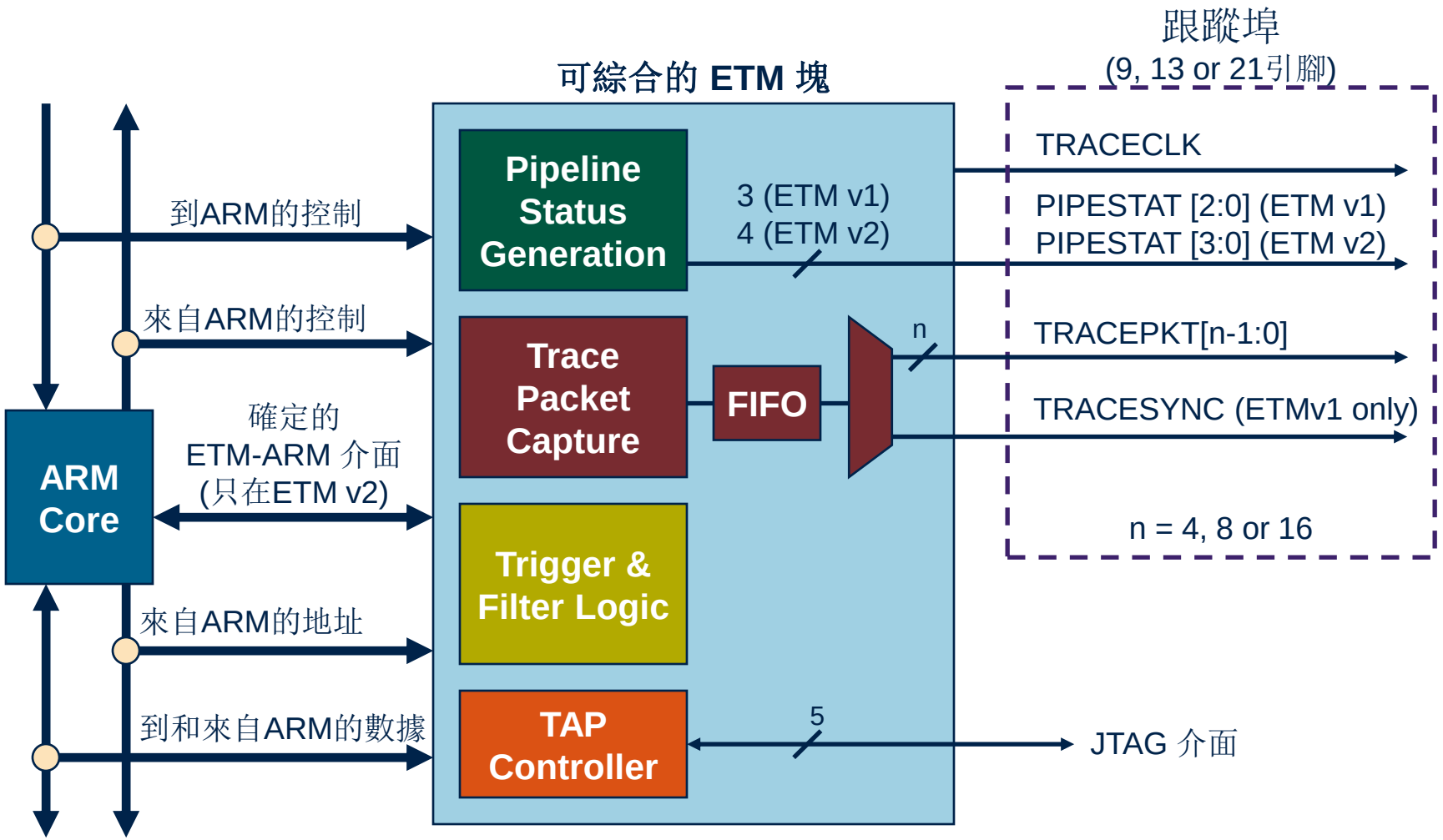


運行跟蹤調試工具的主機
(ADS的附件)

- 嵌入式跟蹤單元 (ETM)
  - 即時指令跟蹤
  - 即時資料訪問跟蹤
  - 包含 ICE功能 (觸發和過濾邏輯)
- MultiTrace 跟蹤埠分析器 (TPA)
  - 深度緩衝器捕獲跟蹤
- 跟蹤調試工具
  - 通過JTAG/Multi-ICE 配置ETM跟蹤
  - 從ETM/MultiTrace裡接收壓縮的跟蹤資料
  - 利用拷貝原代碼映象來對ETM跟蹤

ETM 結構

- ETM 監控器和簡要的ARM 匯流排信號



跟蹤埠說明

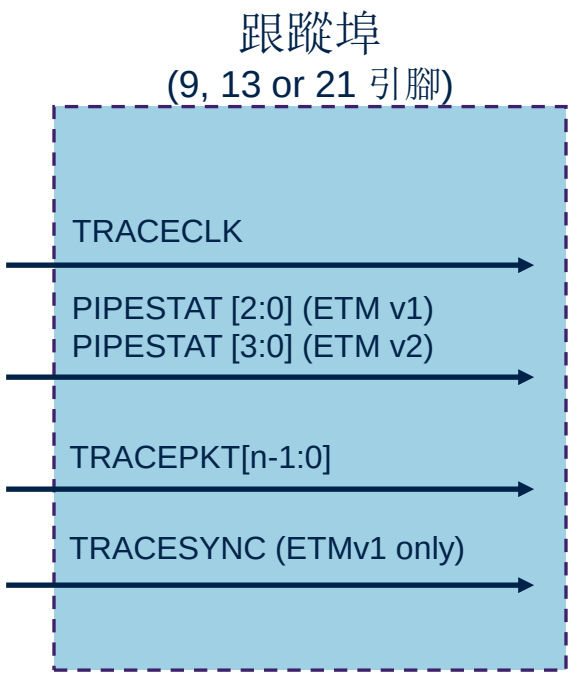
- TRACECLK: **與處理器時鐘有相同的頻率**

- PIPESTAT **(流水線狀態)** 表明:
  - 程式流
  - 是否有一個相關聯的TRACEPKT
  - ETM 狀態

- TRACEPKT **(跟蹤包)** 可能包含以下一個或多個特徵**:**
  - 資料位址
  - 數據值
  - 指令位址

- TRACESYNC **(只在ETM v1有):**
  - 用來在ETM和TPA之間進行同步

*** 可以參考ETM規範得到更多的資訊**



指令跟蹤

- **與指令跟蹤有關的跟蹤埠部分:**
  - PIPESTAT –表明內核的流水線狀態 (例如，一個指令是否被執行)
  - TRACEPKT – 當需要時，包含一個跳轉目標位址
  - 以上兩條和代碼映象相結合，就可以進行調試了

- **指令跟蹤能夠被高度壓縮**
  - 典型地，一個 9-位的跟蹤埠可以處理只有指令的跟蹤
  - ETM v1 最好的情形: 3位元的跟蹤資訊來跟蹤32位元的代碼
  - ETM v2 最好的情形: 4位元的跟蹤資訊來跟蹤64位元的代碼

- **可以用過濾和觸發器:**
  - 只對記憶體位址和/或區域進行跟蹤
  - 只有在特定的ETM順序狀態，計數器等才進行跟蹤

- **技術注解：TraceEnable (一個 ETM 內部信號) 可以激發指令跟蹤。它是由觸發器/篩檢程式事件和資源控制的。**

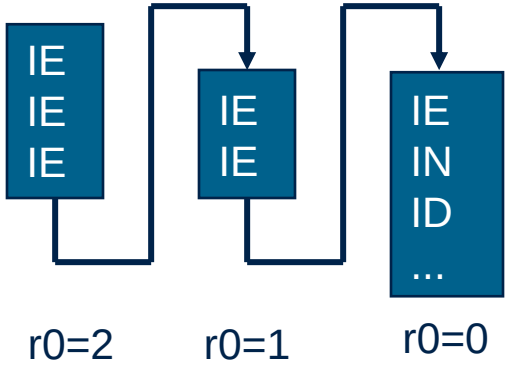
資料跟蹤

- **與資料跟蹤有關的跟蹤埠部分:**
  - TRACEPKT – 包含資料位址或資料值
  - 只有資料位址改變的位元被廣播 (目的是節省頻寬)
- **每一次跟蹤運行都可以配置 ETM:**
  - 資料位址或資料值（或兩個都有）
  - 讀或寫（或兩個都有）
- **需要一個高的頻寬跟蹤所有資料(有程式相關)**
  - 一個帶有45個位元組FIFO的13-位跟蹤埠可以跟蹤大部分資料傳輸
  - 如果想跟蹤更多資料訪問則需要一個21-位元的跟蹤埠
- **篩檢程式和觸發器對保持跟蹤資料的管理很重要**
  - 如果在跟蹤資訊裡有一個 FIFO 溢出標記，那麼就表示有部分的跟蹤資訊被丟掉
  - 推薦使用可程式設計的篩檢程式和觸發邏輯
- **技術注解：ViewData (一個ETM的內部信號) 激發了資料跟蹤(如果TraceEnable是假時被忽略掉). 它是由觸發器/過濾事件和資源控制的。**

跟蹤例子

- 基本指令和直接跳轉的**ETM**跟蹤

```
0x1010   MOV   r0, #3
0x1014   SUBS  r0, r0, #1
0x1018   BNE   0x1014
0x101C   LDR   r1, #0x4000
...
0x4000   0x4321 {data}
```



由PIPESTAT=IE 或 IN來解碼**直接跳轉**

PIPESTAT	TRACEPKT	Corresponding Instruction	Comment
IE	none	0x1010 MOV r0, #3	r0=3
IE	none	0x1014 SUBS r0, r0, #1	r0=2
IE	none	0x1018 BNE 0x1014	direct branch taken
IE	none	0x1014 SUBS r0, r0, #1	r0=1
IE	none	0x1018 BNE 0x1014	direct branch taken
IE	none	0x1010 SUBS r0, r0, #1	r0=0
IN	none	0x1018 BNE 0x1014	branch not taken
ID†	r1†	0x101C LDR r1, #0x4000	

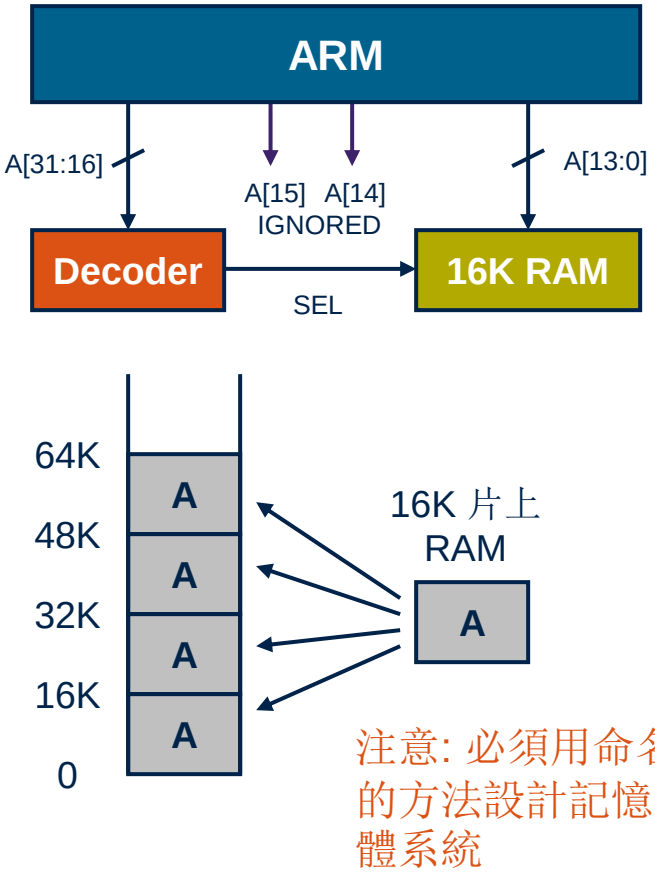
† 假設資料跟蹤是啟動的(**ViewData** 被聲明了)

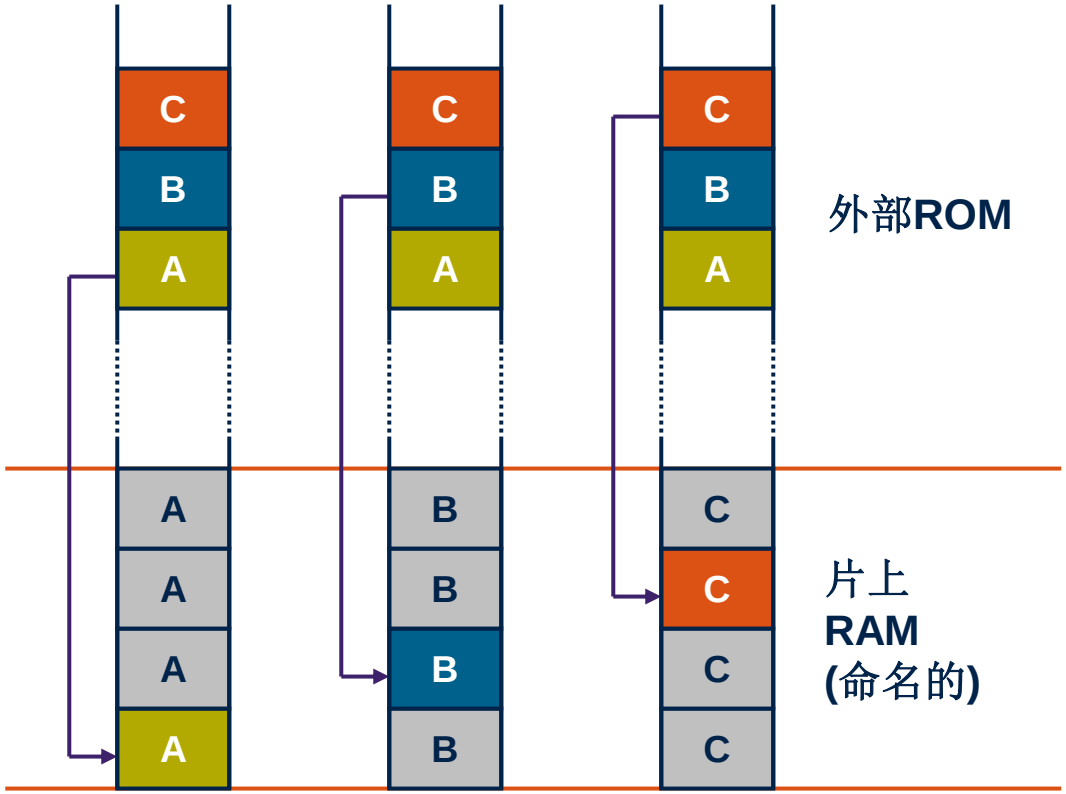
指令跟蹤局限

- **調試器需要一個代碼的拷貝來做參考**
  - 自動修改代碼部分將不能和ETM一起工作
- **調試器必須知道代碼映象的位址映射**
  - 代碼必須在連接時的位址執行
  - 帶有固定代碼位址的虛擬記憶體配置是可以的
  - 現在不支援動態重定位代碼
- **請注意“覆蓋” (例如：把代碼調入片上RAM)**
  - 如果有很多程式碼片段運行在相同的位址，跟蹤不能區分哪一部分在運行。這個例子裡，跟蹤解壓是不可能的。
  - 解決辦法是給記憶體區域命名 (見下一頁)

重疊和命名

- 硬體的命名將每一個段都映射到相同的實際記憶體上
- 用不同的“別名”連接每一個重疊的段執行
- PC值標識了哪一個段在執行





控制跟蹤

- **ETM裡的資源是可以被過濾和控制的：**
  - 要跟蹤的指令
  - 要跟蹤的資料訪問
  - 外部跟蹤埠分析器的觸發

- **ETM的資源由以下部分組成：**
  - 8 個資料比較器
  - 8 對全範圍地址比較器
  - 16個地址解碼器
  - 4 個16-位元數目器
  - 1 個3-狀態序列器
  - 對於ASIC: 最多 4用戶輸入，4個用戶輸出

- **一個事件可以由任何兩個資源邏輯組成：例如**
  "within address range 1 AND data value equals 0xFFFFFFFF"

- **不同的ETM 實現有不同數量的觸發資源**
  - 可以看技術文檔得到更詳細的資料.

哪一個ETM?

- 根據你的ARM內核來選擇 ETM7, ETM9, 或者 ETM10
- 選擇小的，中等的或大的模式
- 選擇跟蹤埠的寬度 (4/8/16)
  - 如果高的吞吐率（資料跟蹤）很重要的時候，可以選擇更寬的埠
  - 可以考慮與其它引腳的多工技術（例如GPIO)

- 調試時可以使用大的寬的，產品時可以使用簡單的，窄的ETM
- 使用ATPG 和插入掃描來對產品進行測試
- ETM 提供了驗證環境套件

- 下一頁將總結配置和資料的大小

ETM 配置

功能	小的	中等的	大的
Addr Comparators	2	8	16
Data Comparators	0	2	8
Range Comparators	1	4	8
Addr Decoders	4	8	16
Sequencers	0	1	1
Counters	1	2	4
Ext. Inputs	2	4	4
Ext. Outputs	0	1	4

ETM v1 (ETM7,9)	小的	中等的	大的
Gate count	25k	35k	60k
Port width	4 / 8	4 / 8 /16	4 / 8 /16
FIFO Depth	9 or 10	18 or 20	45

ETM v2 (ETM10)	小的	中等的	大的
Gate count	35k	50k	75k
Port width	4 / 8 /16	4 / 8 /16	4 / 8 /16
FIFO Depth	15	30	60

實現跟蹤 (1)

ETM7
ARM7TDMI ARM7TDMI-S ARM720T rev3 ARM7EJ

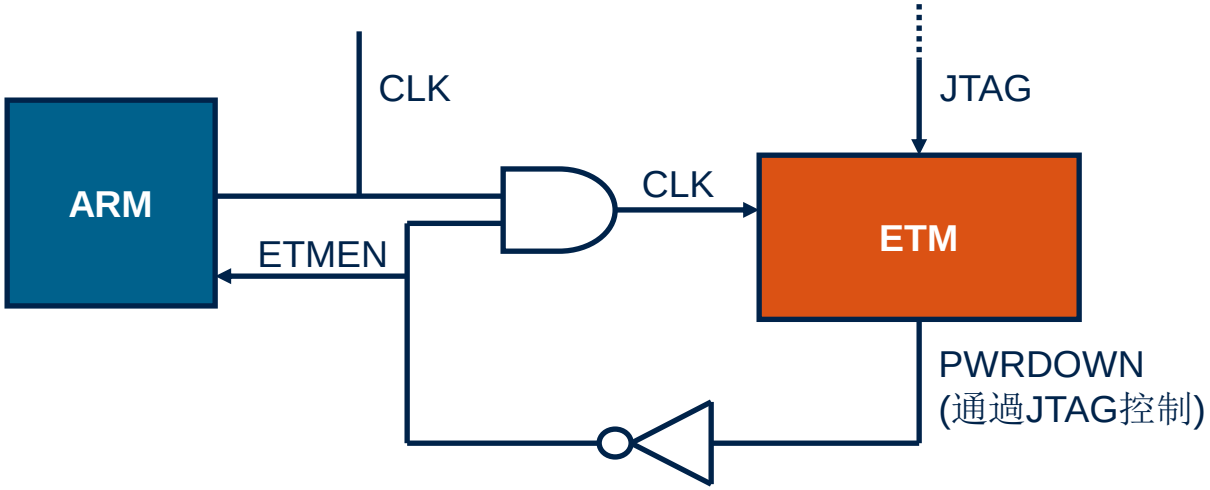
ETM9
ARM9TDMI ARM9E-S/9EJ-S ARM920T rev1/922T ARM966E-S ARM946E-S ARM926EJ-S

ETM10
ARM1020E

- **所有需要的信號連接到ARM 核**
  - 沒有緩存的內核與位址/資料匯流排和控制信號相連
  - 有緩存的內核輸出內核信號到巨集單元邊緣 (例如： ARM920T 版本1)
- **ASIC上提供跟蹤埠**
  - 經過基座上輸出的最高頻率是多少?
  - ASIC上要支持多個ARM內核嗎?
  - 考慮跟蹤埠與別的引腳多工嗎?
  - 最終產品中去除以減少引腳數量嗎?
- **ARM 跟蹤捕獲和JTAG的標準連接器**
  - 38-路的 AMP MICTOR 連接器 (高密度)

實現跟蹤 (2)

- **對 ARM7 和 ARM9 內核:**
  - CPU 時鐘應該自由運行 (利用 nWAIT, 而不是時鐘延長), 但不是至關重要的
- **FIFO FULL 信號可以被用來停止處理器**
  - 但是如果FIFO 滿了時，將影響即時性能.
- **ETM設計的目標不是低功耗**
  - 在正常運行操作（沒有跟蹤）模式下，ETM不應該被時鐘驅動



- **更多更詳細的資訊請看:**
  - ETM 規範
  - ETM7/9/10 技術參考手冊 (TRM's)

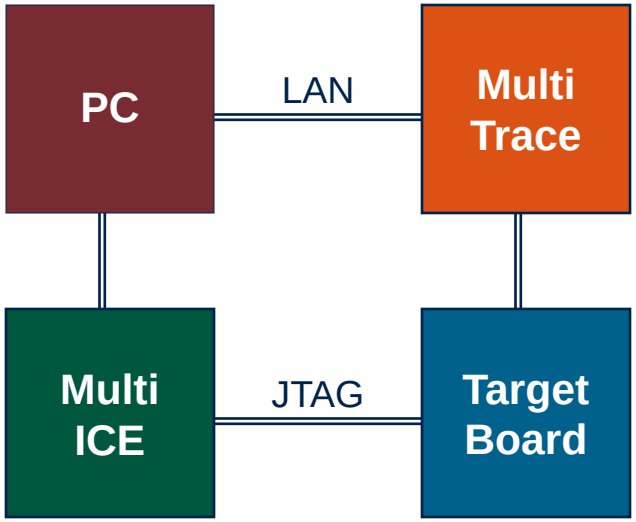
跟蹤捕獲

- **需要一個外部的跟蹤埠分析儀(TPA)來捕獲 ETM 的輸出**
  - TPA's 可以在觸發事件之前，發生時或之後來跟蹤
  - TPA's 帶有很深的跟蹤捕獲緩衝器
  - TPA 的緩衝器通過很高速度的上行連接（通常是乙太網）來把資料返回給主機 PC
- **當前可選擇的TPA :**
  - Multi-Trace**
  - Agilent 邏輯分析儀 (需要 Multi-ICE 或者 Agilent JTAG 探測器)
  - Agilent ARM(E5904B)的跟蹤埠分析儀 `http://www.tm.agilent.com/`
  - Tektronix 邏輯分析儀**
    - `http://www.tek.com/`

** 需要 Multi-ICE 2.0 或 更新的

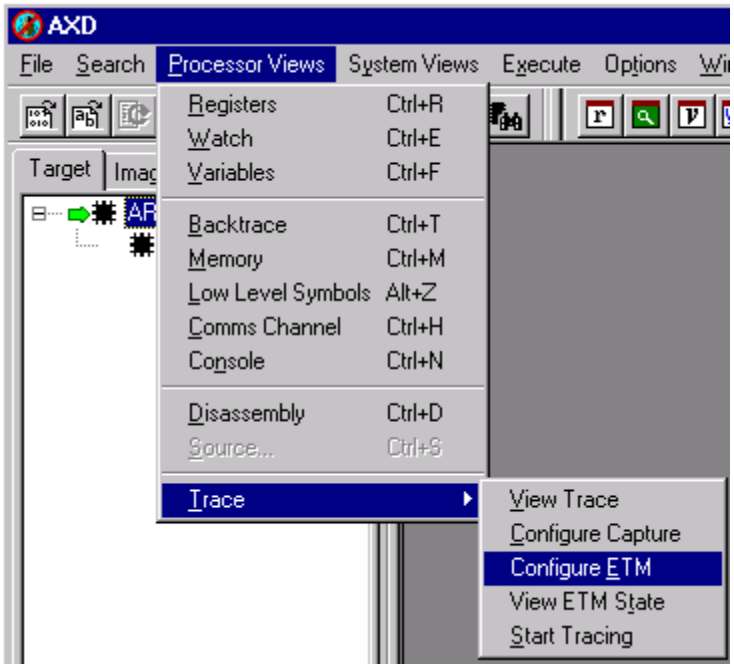
- **Lauterbach 也提供跟蹤工具、模擬器、調試器**
  - `http://www.lauterbach.com/`

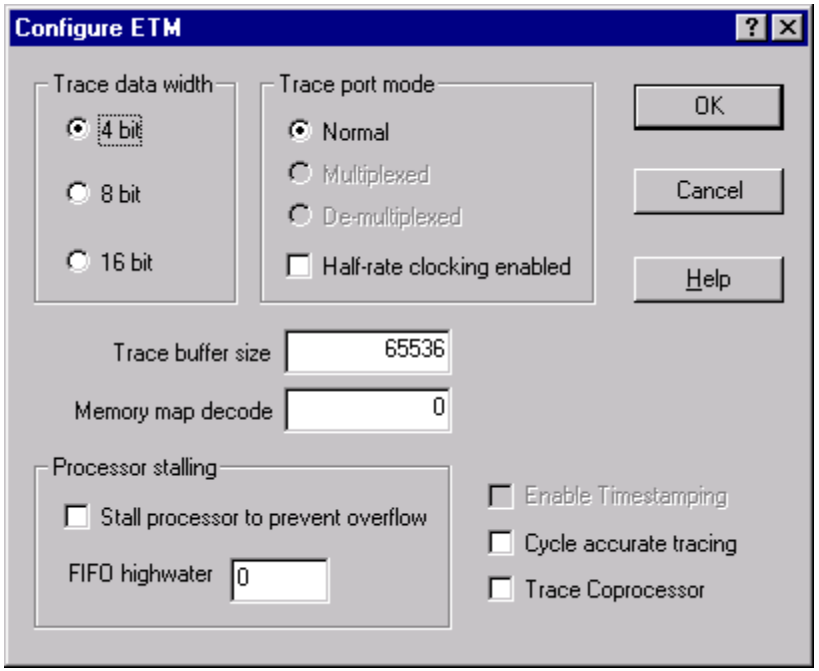
ARM MultiTrace



- **與 Multi-ICE (2.0+)連接一起工作**
  - 通過乙太網跟蹤，通過JTAG口控制運行
  - 支持 9, 13 和 21 引腳的跟蹤埠，頻率可高達 200 MHz
  - 1.0V - 3.3V目標電壓
  - 支持將跟蹤時脈速率降低一半，在上升和下降沿捕獲資料
- **也可以從協力廠商那獲得跟蹤分析儀方案**
  - **訪問：** www.arm.com

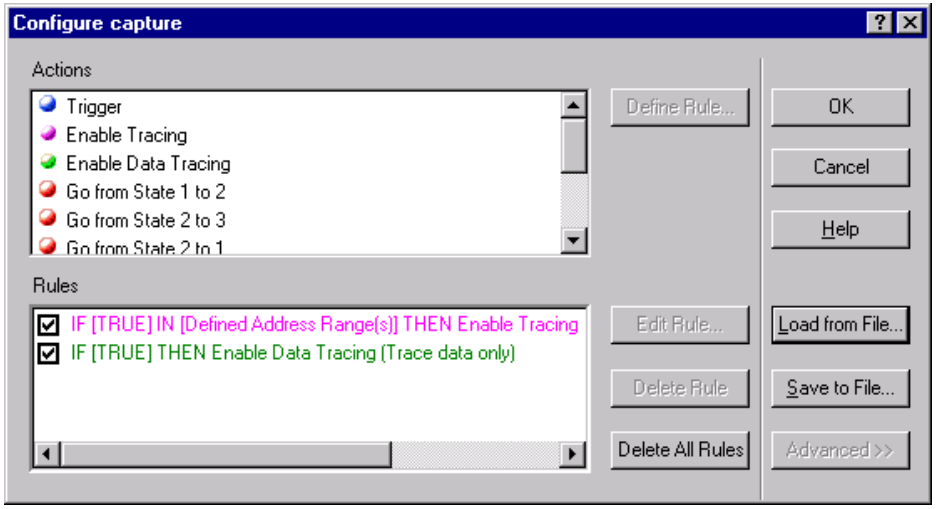
跟蹤調試工具概況

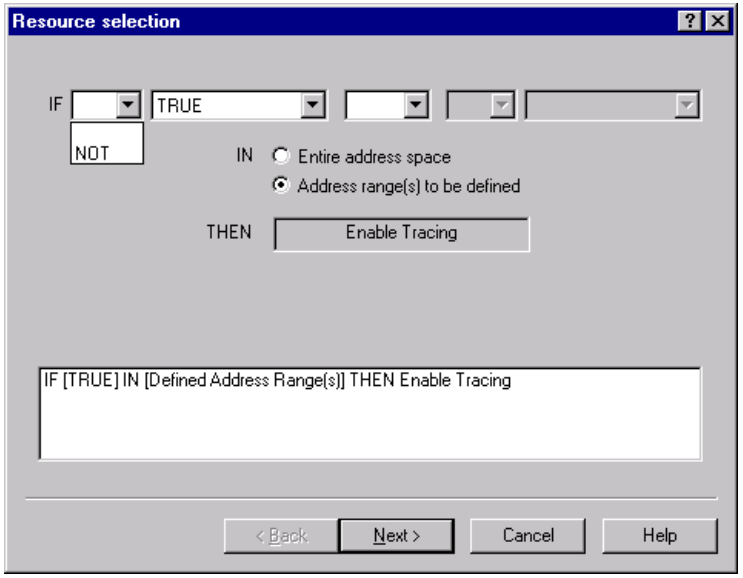




- **ADS 1.1/1.2- TDT 1.1.1 作為附加工具加到 AXD裡**
  - 有額外的使用者介面功能來支援跟蹤
  - 通過JTAG配置 ETM
  - 收集跟蹤資訊
  - 解碼跟蹤資訊

跟蹤配置





- **ETM 寄存器設置的 GUI 介面**
  - 對所有可能的跟蹤和觸發ETM配置進行全權訪問
  - 適合於已實現的資源對話視窗

- **存儲/恢復 觸發器的例子**

跟蹤輸出

Trace

ARM966E_S_ETM(L) - Trace Sync: Off Tracing: On

Index	Address	Data	Mnemonic	Status	ARM/Thumb
-43	0x0000811c	0xE1540000	cmp r4,r0		ARM
-41	0x00008120	0x1A000002	bne 0x8130 ; (main + 0x88)		ARM
	27	glob.c	localvar++;		
-37	0x00008130	0xE2840001	add r0,r4,#1		ARM
-36	0x00008134	0xE1A04000	mov r4,r0		ARM
-35	0x00008138	0xEAFFFFEC	b 0x80f0 ; (main + 0x48)		ARM
	20	glob.c	while (localvar != 2000)		
-34	0x000080f0	0xE3540E7D	cmp r4,#0x7d0		ARM
-33	0x000080f4	0x0A000010	beq 0x813c ; (main + 0x94)	Not executed	ARM
	22	glob.c	*pointer++ = localvar;		
-31	0x000080f8	0xE5854000	str r4,[r5,#0]		ARM
-31		0x000005DC	<Word Write>		
-29	0x000080fc	0xE2855004	add r5,r5,#4		ARM
	23	glob.c	if (localvar == start)		
-26	0x00008100	0xE59D0008	ldr r0,[r13,#8]		ARM
-26		0x0000044C	<Word Read>		
-24	0x00008104	0xE1540000	cmp r4,r0		ARM
-22	0x00008108	0x1A000002	bne 0x8118 ; (main + 0x70)		ARM
	25	glob.c	if (localvar == stop)		
-17	0x00008118	0xE59D0004	ldr r0,[r13,#4]		ARM

- 映射回執行窗口
  - 產生交叉源碼察看

問題

1) 跟蹤需要多少個引腳?

2) 跟蹤輸出應該使用什麼樣的物理連接器?

3) 跟蹤埠(TRACECLK)的速度是多少?

4) 大量的資料跟蹤需要什麼大小的跟蹤埠?

5) 怎麼更好地減少跟蹤引腳的多少?

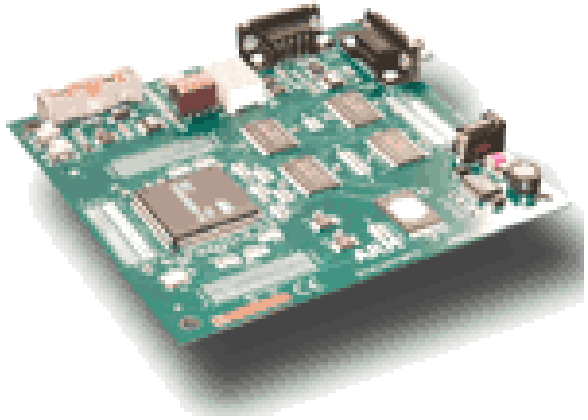
6) 什麼類型的程式不能被跟蹤?

7) 如果需要覆蓋跟蹤，應該需要什麼樣的硬體功能?

議程

- **基本調試需求**
  - 你需要什麼樣的功能？
  - ARM公司的調試和開發組成工具。
- **嵌入式核調試**
  - 實現和利用JTAG的調試方案
  - 停止模式和監控模式
- **嵌入式跟蹤**
  - 整體化和利用ETM
- **ARM 開發板**

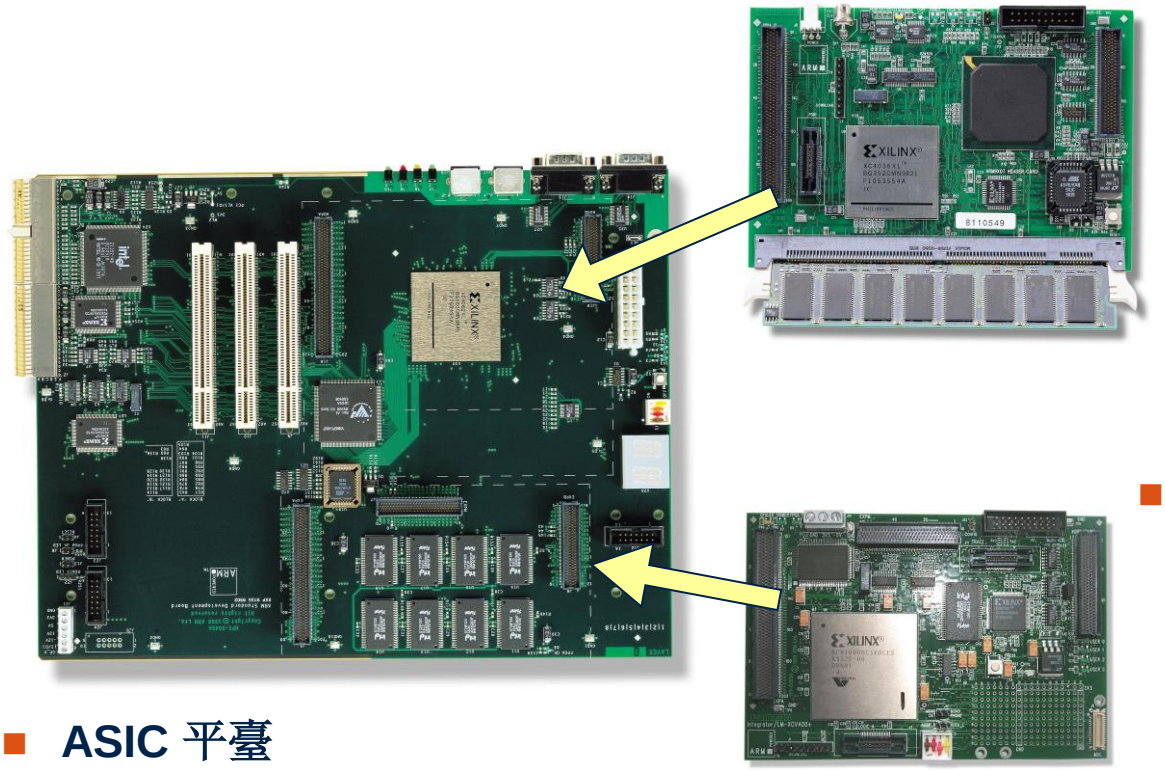
Evaluator-7T



可以從網上購買
www.arm.com

- **ARM技術的簡單介紹，價格低**
- **包括:**
  - Samsung KS32C50100控制器
  - 512k 快閃記憶體, 512k SRAM
  - 2 串口, 7個 LED 顯示
  - 複位和中斷開關
  - Multi-ICE 連接器
  - 板上Angel 調試監控工具
  - ADS評估版
  - 電壓和電纜線

Integrator 平臺



- 核模組
  - ARM 內核
  - 至少 256kb SSRAM
  - 多達 256Mb SDRAM
  - 可以疊 4 層
- 邏輯模組
  - 多達 2M門的FPGA
  - 1Mb SRAM
  - 分析儀/跟蹤 連接器
  - 時鐘發生器
- **ASIC 平臺**
  - 可達5 個核/邏輯 模組
  - 系統匯流排時鐘
  - 32Mb 快閃記憶體
  - FPGA系統控制器
  - AMBA 和PCI 擴展

還有介面模組，
分析器模組和 移植模組
請訪問： **www.arm.com**

參考資料

- **Multi-ICE**
  - Multi-ICE 用戶手冊, 附錄, readme.txt 和 troubleshoot.txt
  - ADS 1.2: AXD 和 armsd 調試器手冊
  - ADS 1.2: 調試目標手冊 (第5章: Semihosting SWIs)
  - 應用注釋 31 : 使用 EmbeddedICE
- **MultiTrace**
  - ETM7, ETM9, ETM10 技術參考手冊
  - 嵌入式跟蹤巨集單元規範
  - MultiTrace 用戶手冊
  - 跟蹤調試工具用戶手冊
- **通用**
  - 你使用的 ARM 內核的調試信息
- **開發板**
  - www.arm.com/


ARM®
THE ARCHITECTURE
FOR THE DIGITAL WORLD™