CHƯƠNG 2 HỆ PHÁT TRIỂN PHẦN MỀM HỆ THỐNG ĐIỀU KHIỂN SỐ

I. ĐẶC TÍNH VÀ NHIỆM VỤ CỦA HỆ PHÁT TRIỂN PHẦN MỀM

Hệ phát triển phần mềm thực hiện các nhiệm vụ sau:

- Cấu hình cho Chip vi xử lý. Phụ thuộc vào họ vi xử lý sử dụng mà việc cấu hình cho Chip khác nhau, việc cấu hình cho Chip được hiểu là cài đặt các tham số cho phép một vài khối phần cứng hoạt động theo sự lựa chọn của người sử dụng. Ví dụ đối với vi xử lý họ AVR có thể là cấu hình phương pháp tạo dao động bên trong Chip, kiểm tra nhiễu tác ddongj qua đường nguồn, bất tắt WatchDog Timer .. Còn đối với họ PSoC là bài toán xây dựng các Module phần cứng bên trong Chip
- Soạn thảo chương trình nguồn
- Biên dịch chương trình nguồn sang dạng mã máy
- Liên kết các chương trình đã biên dịch thành file có thể nạp trực tiếp vào Chip.
- Mô phỏng và gỡ rối chương trình.

Biên dịch

Quá trình biên dịch (*Compiling*)

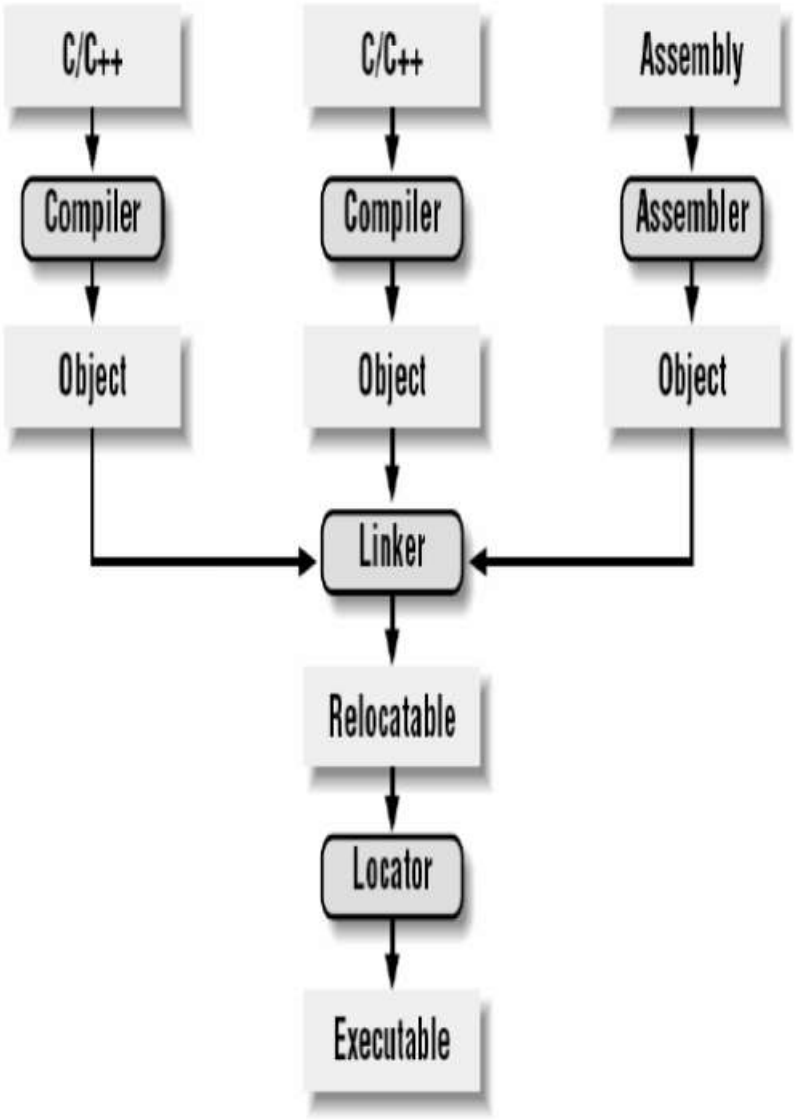
Nhiệm vụ chính của bộ biên dịch là chuyển đổi chương trình được viết bằng ngôn ngữ thân thiện với con người ví dụ như C, C++,…thành tập mã lệnh tương đương có thể đọc và hiểu bởi bộ vi xử lý đích. Theo cách hiểu này thì bản chất một bộ hợp ngữ cũng là một bộ biên dịch để chuyển đổi một-một từ một dòng lệnh hợp ngữ thành một dạng mã lệnh tương đương cho bộ vi xử lý có thể hiểu và thực thi. Chính vì vậy đôi khi người ta vẫn nhầm hiểu giữa khái niệm bộ hợp ngữ và bộ biên dịch. Tuy nhiên việc biên dịch của bộ hợp ngữ sẽ được thực thi đơn giản hơn rất nhiều so với các bộ biên dịch cho các mã nguồn viết bằng ngôn ngữ bậc cao khác.

Mỗi một bộ xử lý thường có riêng ngôn ngữ máy vì vậy cần phải chọn lựa một bộ biên dịch phù hợp để có thể chuyển đổi chính xác thành dạng mã máy tương ứng với bộ xử lý đích. Đối với các hệ thống nhúng, bộ biên dịch là một chương trình ứng dụng luôn được thực thi trên máy chủ (môi trường phát triển chương trình) và còn có tên gọi là bộ biên dịch chéo. Vì bộ biên dịch chạy trên một nền phần cứng để tạo ra mã chương trình chạy trên môi trường phần cứng khác. Việc sử dụng bộ biên dịch chéo này là một thành phần không thể thiếu trong quá trình

phát triển phần mềm cho hệ nhúng. Các bộ biên dịch chéo thường có thể cấu hình để thực thi việc chuyển đổi cho nhiều nền phần cứng khác nhau một cách linh hoạt. Và việc lựa chọn cấu hình biên dịch tương ứng với các nền phần cứng đôi khi cũng khá độc lập với chương trình ứng dụng của bộ biên dịch.

Kết quả đầu tiên của quá trình biên dịch nhận được là một dạng mã lệnh được biết tới với tên gọi là tệp đối tượng (*object file*). Nội dung của tệp đối tượng này có thể được xem như ***là một cấu trúc dữ liệu trung gian*** và thường được định nghĩa như một định dạng chuẩn COFF (*Common Object File Format*) hay định dạng của bộ liên kết mở rộng ELF (*Extended Linker Format*)… Nếu sử dụng nhiều bộ biên dịch cho các *modul* mã nguồn của một chương trình lớn thì cần phải đảm bảo rằng các tệp đối tượng được tạo ra phải có chung một kiểu định dạng.

Hầu hết nội dung của các tệp đối tượng đều bắt đầu bởi một phần *header* để mô tả các phần theo sau. Mỗi một phần sẽ chứa một hoặc nhiều khối mã hoặc dữ liệu như được sử dụng trong tệp mã nguồn. Tuy nhiên các khối đó được nhóm lại bởi bộ biên dịch vào trong các phần liên quan. Ví dụ như tất cả các khối mã được nhóm lại vào trong một phần được gọi là *text*, các biến toàn cục đã được khởi tạo (cùng các giá trị khởi tạo của chúng) vào trong phần dữ liệu, và các biến toàn cục chưa được khởi tạo vào trong phần bss.



Quá trình phát triển và biên dịch phần mềm nhúng

Cũng khá phổ biến thường có một bảng biểu tượng chứa trong nội dung của tệp đối tượng. Nó chứa tên và địa chỉ của tất cả các biến và hàm được tham chiếu trong tệp mã nguồn. Các phần chứa trong bảng này không phải lúc nào cũng đầy đủ vì có một số biến và hàm được định nghĩa và chứa trong các tệp mã nguồn khác. Chính vì vậy cần phải có bộ liên kết để thực thi xử lý vấn đề này.

Quá trình liên kết (*Linking*)

Tất cả các tệp đối tượng nhận được sau bước thực hiện biên dịch đầu tiên đều phải được tổ hợp lại theo một cách đặc biệt trước khi nó được nạp và chạy ở trên môi trường phần cứng đích. Như đã thấy ở trên, bản thân các tệp đối tượng cũng có thể là chưa hoàn thiện vì vậy bộ liên kết phải xử lý để tổ hợp các tệp đối tượng

đó với nhau và hoàn thiện nốt phần còn khuyết cho các biến hoặc hàm tham chiếu liên thông giữa các tệp mã nguồn được biên dịch độc lập.

Kết quả đầu ra của bộ liên kết ***là một tệp đối tượng mới có chứa tất cả mã và dữ liệu trong tệp mã nguồn và cùng kiểu định dạng tệp***. Nó thực thi được điều này bằng cách tổ hợp một cách tương ứng các phần *text*, dữ liệu và phần *bss* …từ các tệp đầu vào và tạo ra một tệp đối tượng theo định dạng mã máy thống nhất. Trong qúa trình bộ liên kết thực hiện tổ hợp các phần nội dung tương ứng nó còn thực hiện thêm cả vấn đề hoàn chỉnh các địa chỉ tham chiếu của các biến và hàm chưa được đầy đủ trong bước thực hiện biên dịch.

Các bộ liên kết có thể được kích hoạt thực hiện độc lập với bộ biên dịch và các tệp đối tượng được tạo ra bởi bộ biên dịch được coi như các tham biến vào. Đối với các ứng dụng nhúng nó thường chứa phần mã khởi tạo đã được biên dịch cũng phải được gộp ở trong danh sách tham biến vào này.

Nếu cùng một biểu tượng được khai báo hơn một lần nằm trong một tệp đối tượng thì bộ liên kết sẽ không thể xử lý. Nó sẽ kích hoạt cơ chế báo lỗi để người phát triển chương trình xem xét lại. Hoặc khi một biểu tượng không thể tìm được địa chỉ tham chiếu thực trong toàn bộ các tệp đối tượng thì bộ liên kết sẽ cố gắng tự giải quyết theo khả năng cho phép dựa vào các thông tin ví dụ như chứa trong phần mô tả của thư viện chuẩn. Điều này cũng thường hoặc có thể gặp với trường hợp các hàm tham chiếu trong chương trình.

Rất đáng tiếc là các hàm thư viện chuẩn thường yêu cầu một vài thay đổi trước khi nó có thể được sử dụng trong chương trình ứng dụng nhúng. Vấn đề ở đây là các thư viện chuẩn cung cấp cho các bộ công cụ phát triển chỉ dừng đến khả năng định dạng và tạo ra tệp đối tượng. Hơn nữa chúng ta cũng rất ít khi có thể truy nhập được vào mã nguồn của các thư viện chuẩn để có thể tự thay đổi. Hiện nay cũng có một số nhà cung cấp dịch vụ phần mềm hỗ trợ công cụ chuyển đổi hay thay đổi thư viện C chuẩn để ứng dụng cho các ứng dụng nhúng, ví dụ như *Cygnus*. Gói phần mềm này được gọi là *newlib* và được cung cấp miễn phí. Chúng ta có thể tải về trang web của *Cygnus*. Nó sẽ hỗ trợ chúng ta giải quyết vấn đề mà bộ liên kết có thể gặp phải khi chương trình sử dụng các hàm thuộc thư viện C chuẩn.

Sau khi đã hợp nhất thành công tất cả các thành phần mã và phần dữ liệu tương ứng cũng như các vấn đề về tham chiếu tới các biểu tượng chưa được thực thi

trong quá trình biên dịch đơn lẻ, ***bộ liên kết sẽ tạo ra một bản sao đặc biệt của chương trình có***
khả năng định vị lại (relocatable). Hay nói cách khác, chương trình được hoàn thiện ngoại trừ một điều: Không có địa chỉ bộ nhớ nào chưa được gán bên trong các phần mã và dữ liệu. Nếu chúng ta không phải là đang phát triển phần mềm cho hệ nhúng thì uá trình biên dịch có thể kết thúc tại đây. Tuy nhiên, với hệ nhúng ngay cả hệ thống nhúng đã bao gồm cả hệ điều hành chúng ta vẫn cần phải có một mã chương trình (*image*) nhị phân được định vị tuyệt đối. Thực tế nếu có một hệ điều hành thì phần mã và dữ liệu cũng thường gộp cả vào bên trong chương trình có khả năng định vị lại.

Toàn bộ ứng dụng nhúng bao gồm cả hệ điều hành thường liên kết tĩnh với nhau và thực hiện như một mã chương trình nhị phân thống nhất.

Quá trình định vị (*Locating*)

Công cụ thực hiện việc chuyển đổi một chương trình có khả năng định vị lại thành một dạng mã chương trình nhị phân có thể thực thi được gọi là bộ định vị. Nó sẽ đảm nhiệm vai trò của bước đơn giản nhất trong các bước thực thi biên dịch nói chung. Thực tế chúng ta phải tự làm hầu hết công việc của bước này bằng cách cung cấp thông tin về bộ nhớ đã được cấu hình trên nền phần cứng mà chúng ta đang phát triển và đó chính là tham số đầu vào cho việc thực thi của bộ định vị. Bộ định vị sẽ sử dụng thông tin này để gán các địa chỉ vật lý cho mỗi phần mã lệnh và dữ liệu bên trong chương trình được thực thi mà có thể định vị lại. Tiếp theo nó sẽ tạo ra một tệp có chứa chương trình bộ nhớ nhị phân để có thể nạp trực tiếp vào bộ nhớ chương trình trên nền phần cứng thực thi.

Trong nhiều trường hợp bộ định vị là một chương trình khá độc lập với các phần công cụ khác trong hệ thống phần mềm phát triển. Tuy nhiên trong các bộ công cụ phát triển GNU chức năng này được tích hợp luôn trong bộ liên kết. Tuy nhiên không nên nhầm lẫn về chức năng của chúng trong quá trình thực thi biên dịch. Thông thường chương trình chạy trên các máy tính mục đích chung thì hệ điều hành sẽ thực hiện việc chuyển đổi và gán chính xác địa chỉ thực cho các phần mã và dữ liệu trong chương trình ứng dụng, còn với chương trình phát triển chạy trên hệ nhúng thì việc này phải được thực hiện bởi bộ định vị. Đây cũng chính là điểm khác biệt cơ bản khi thực hiện biên dịch một chương trình ứng dụng cho hệ nhúng.

Thông tin về bộ nhớ vật lý của hệ thống phần cứng phát triển mà cần phải cung cấp cho bộ định vị GNU phải được định dạng theo kiểu biểu diễn của bộ liên kết. Thông tin này đôi khi được sử dụng để điều khiển một cách chính xác thứ tự trong các phần mã chương trình và dữ liệu bên trong chương trình có thể định vị lại. Nhưng ở đây chúng ta cần phải thực hiện nhiều hơn thế, tức là phải thiết lập chính xác khu vực của mỗi phần trong bộ nhớ.

Ví dụ với một hệ phát triển đối với chip PSoC

PSoC Designer chạy trên hệ điều hành Window tích hợp với môi trường phát triển (IDE) với các chip PSoC. Với môi trường phát triển này người thiết kế có thể tạo cấu hình cho chip, viết mã chương trình ứng dụng và gỡ rối chương trình. PSoC Designer hỗ trợ lập trình trên ngôn ngữ bậc cao C.

Trong môi trường phát triển này gồm có các phần chính sau:

* Thiết kế chip

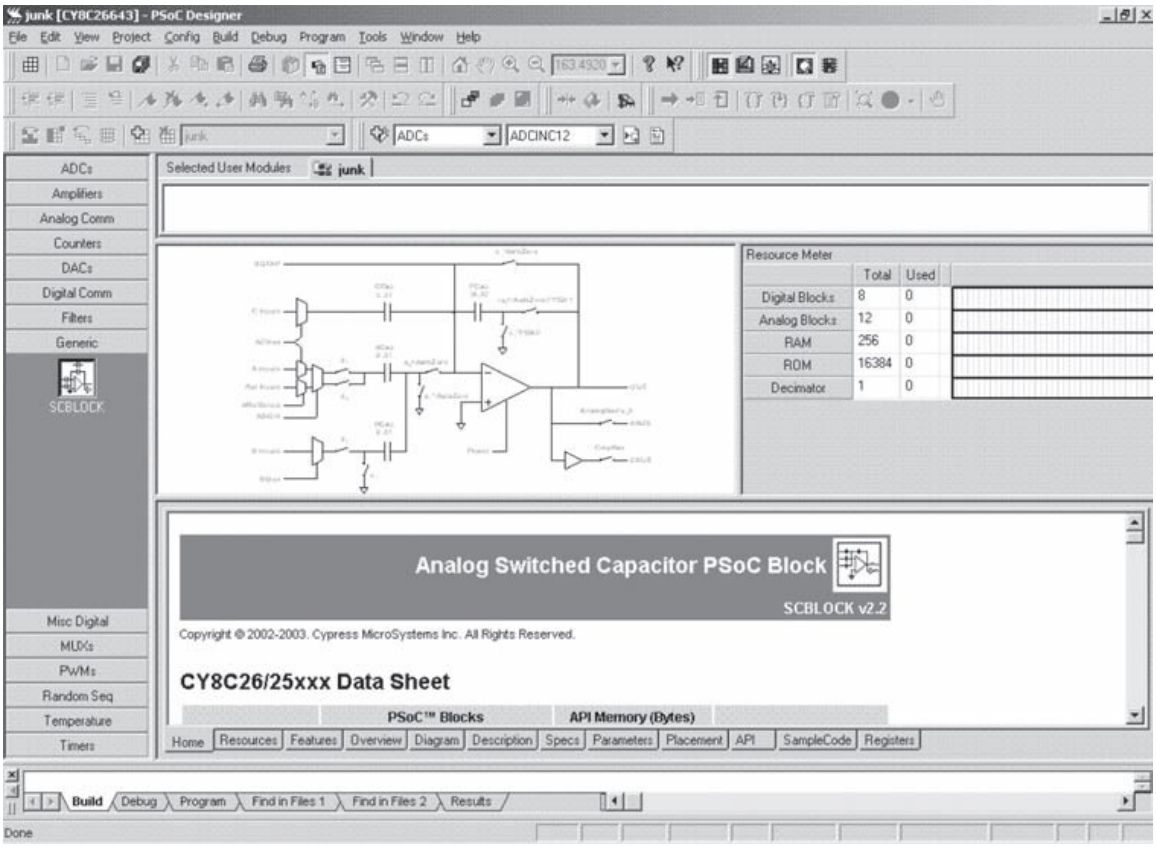
Trong phần này ta có thể dễ dàng tạo cấu hình cho chip và tương ứng với các khôi tạo ra chương tình sẽ tự động tạo ra module phần mềm ứng dụng phù hợp. Khi thiết kế chip, ta có thể lựa chọn các module đã có sẵn và đặt vị trí tương ứng bên trong chip. Với chip PSoC cho phép thiết kế với đa cấu hình và khả năng lựa chọn cấu hình làm việc hiện tại vào bất kỳ thời điểm nào. Viết thiết kế chip được thực hiện dễ dàng nhờ 1 bảng thuộc tính lựa chọn các thông số có sẵn và tự dao động mà nguồn làm việc tương ứng với module lựa chọn.

Phần thiết kế Chip chia thành 2 phần: phần lựa chọn các Module và phấn kết nối các Module với nhau và ra đầu ra.

Lựa chọn các Module người sử dụng như trong hình 2.1

Ở bên phía trái có một danh sách các module có thể lựa chọn, các thông tin về các đặc điểm kỹ thuật của module được trình bày theo sơ đồ phía dưới. Các module được sử dụng có thể là ADC, bộ khuyếch đại tương tự, khối truyền thông tương tự, bộ đếm, khối chuyển đổi số tương tự, khối truyền thông số, bộ lọc, các khối logic cơ bản, bộ dồn kênh, khối điều chế độ rộng xung, khối tạo dữ liệu ngẫu nhiên, khố nhiệt độ và bộ định thời.

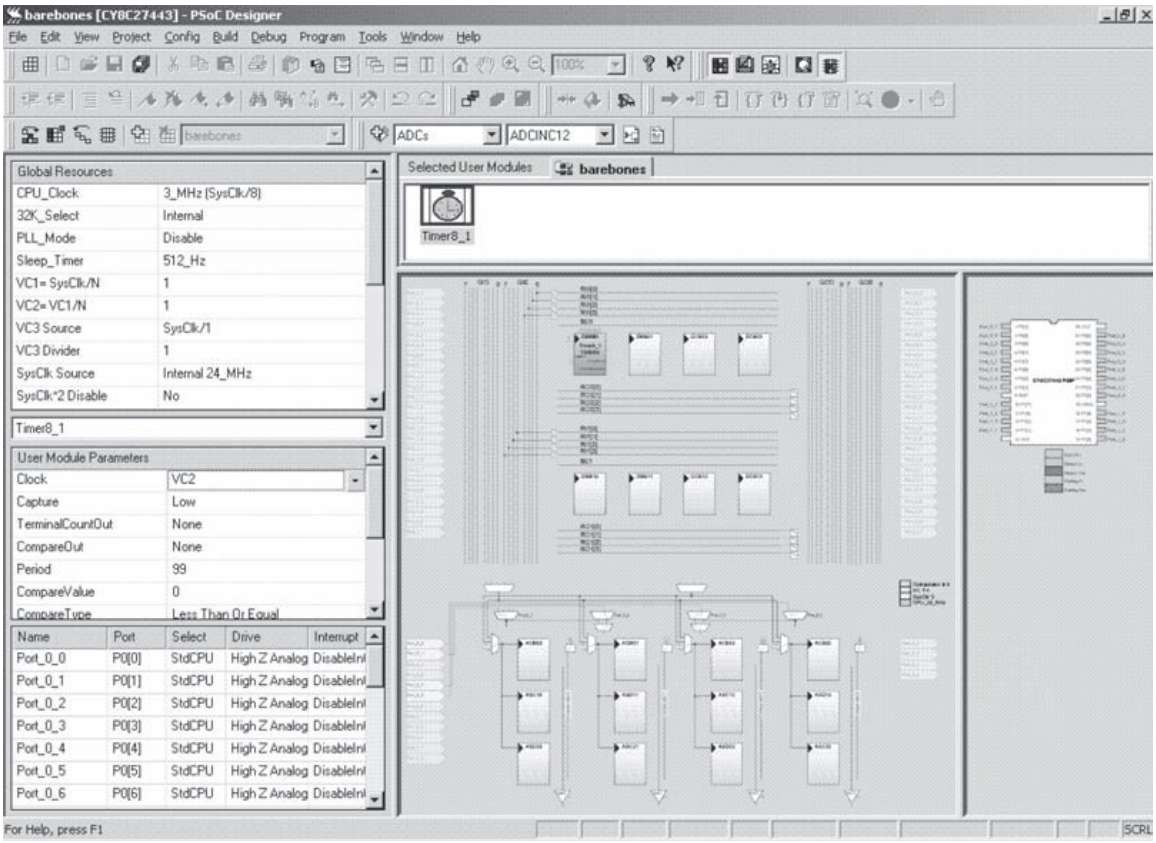
Mỗi khối đều được mô tả các đặc tính kỹ thuật và cách sử dụng trong phần Datasheet. Khi sử dụng các khối sẽ chiếm một khoản tài nguyên của hệ thống (Resource), bảng sử dụng tài nguyên nằm ở phía phải. Bao gồm tài nguyên khối Digital, Analog, RAM, ROM



Hình 2.1 Lựa chọn các module

Phần thiết kế thứ 2 thể hiện sơ đồ kết nối toàn bộ phần cứng bên trong Chip. Trong sơ đồ này cho phép:

- Cài đặtcác tài nguyên chung của Chip như điện áp làm việc, tần số dao động, bộ chia tần ...
- Đặt và cấu hình cho từng module.
- Thiết lập các đường nối giữa các khối và đầu ra.
- Cấu hình các cổng vào ra.



Hình 2.2 Sơ đồ kết nối

* Thiết kế ứng dụng

Trong phần thiết kế ứng dụng, ta có thể soạn thảo chương trình trên ngôn ngữ C hoặc Assembler . Sau đó có thể biên dịch, liên kết chương trình nguồn và tạo ra file mã máy sẵn sàng nạp vào Chip.

- Biên dịch Assembler : Cho phép biên dịch code Assembler két hợp và mã chương trình trên ngôn ngữ bậc cao C thành dạng mã máy. Việc liên kết các thư viện tự động cấp phát địa chỉ tuyệt đối và quản lý các biến bên trong bộ nhớ.

Biên dịch C : Hệ phát triển hỗ trợ chương trình biên dịch C theo chuẩn ANSI (loại trừ dạng dữ liệu 64 bit double). Ngoài ra hỗ trợ hệ thư viện nhưng để điều khiển vào port, hỗ trợ các hàm quét phím và hiển thị màn hình, các hàm toán học

*Trình duyệt thiết kế .

Cho phép đưa vào thiết kế hiện đại các thiết kế chíp đã xây dựng trước đó , điều khiển này giảm thời gian thiết kế và tạo ra sự kế thừa các mã mẫu thiết kế .

Quá trình thiết kế chip PSoC khác về cơ bản so với thiết kế với Chip vi xử lý truyền thống. Tính mềm dẻo về mặt cấu trúc của PSoC thể hiện ở khả năng cấu

hình cho các khối phần cứng (analog và Digital) bên trong .Như trên ta đã thấy bên trong chip PSoC có các khối : Analog và Digital (với loại chip CY8C276443 có 12 khối analog và 8 khối Digital), Ta có thể hiểu các khối này như các vùng được định dạng sẵn cho một chức năng sử dụng nào đó .Ví dụ đối với các khối Digital sẽ có hai loại định dạng là khối Digital cơ sở (DB) chỉ cho phép đặy vào nó các khối như timer , counter, PWM, tạo giá trị ngẫu nhiên (PRS) hay các CRC. Còn Module truyền thông cho phép đặt vào nó ngoài các khối trên còn các khối phục vụ truyền thông như truyền không đồng bộ UART, truyền đồng bộ SPI chế độ Master và Slave. Mỗi một khuôn dạng có dung lượng dữ liệu là 8 bit với các khối phần cứng dung lượng dữ liệu lớn hơn phải dùng nhiều hơn 1 khuôn định dạng

Đối với các module Analog có ba dạng định dạng là khối định dạng tín hiệu liên tục (Continous Time – CT) khối định dạng chuyển mạch tụ loại c và khối chuyển mạch tụ loại D (Switch Capacitor SCC và SCD) .Với định dạng loại CT dùng cho các chức năng làm việc với tín hiệu tương tự liên tục.

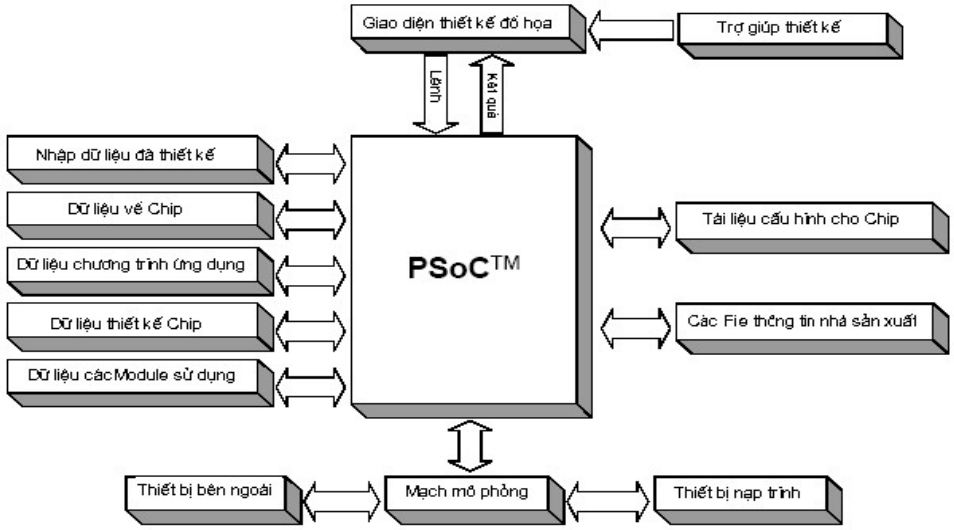
Ví dụ như các bộ khuyếch đại tuyến tính chọn kênh analog. Còn với dạng loại SC dùng cho các tín ứng dụng như bộ chuyển đổi ADC, biến đổi DAC, bộ lọc, mạch trích mẫu và giữ , các bộ so sánh analog và sensor đo nhiệt độ. Khi thiết kế chip vẫn lựa chọn các khối chức năng cần thiết và đặt vào các khuôn dạng phù hợp với nó bên trong chíp, mỗi khuôn định dạng là nguồn tài nguyên của Chip, trong trường hợp với ứng dụng ta vẫn cần sử dụng nhiều hơn tài nguyên của chíp cần thực hiện những giải pháp sau:

- Trước tiên cần tối ưu hoá việc thiết kế Chip, với những người thiếu kinh nghiệm trong quá trình thiết kế có thể sử dụng lãng phí tài nguyên, trong nhiều trường hợp có khai báo sử dụng, nhưng trong quá trình làm việc hầu như không sử dụng, hoặc hiệu quả sử dụng không cao. Hoặc có thể dùng khối khác hiện tại đang dùng có thể thay thế được. Do đó vẫn phải tối ưu hoá trong quá trình thiết kế.

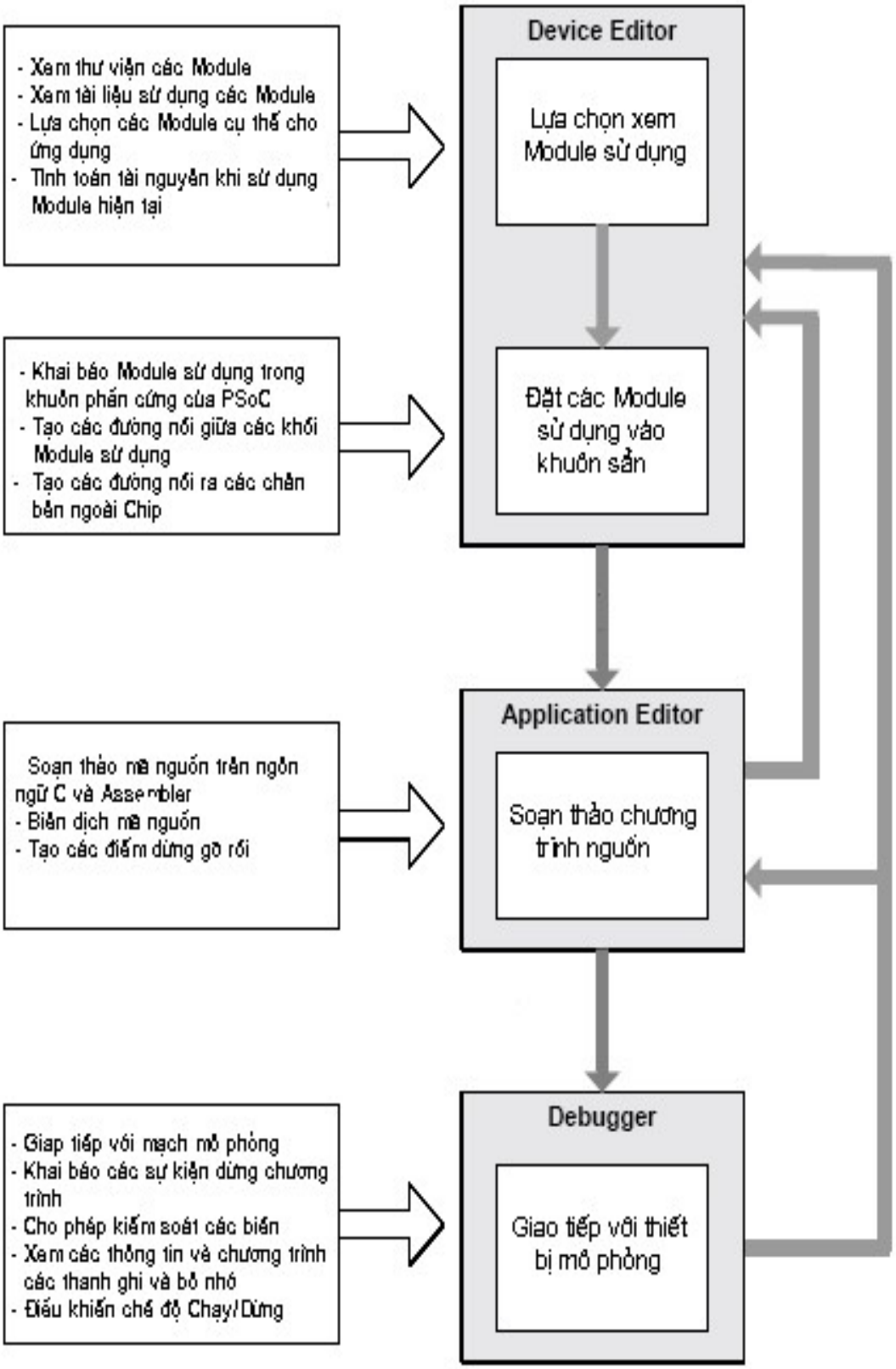
- Trong nhiều trường hợp các khối chức năng không hoạt động cùng một thời điểm, ta có thể sử dụng giải pháp thiết kế nhiều cấu hình cho chíp và sử dụng chế độ nạp cấu hình động phù hợp với quá trình điều khiển thích hợp . Đây là tính năng vượt trội của PSoC so với các loại Chip khác, nó cho phép tận dụng tối đa tài nguyên bên trong Chip.

- Cần lựa chọn loại chip có nhiều khuôn định dạng bên trong Chip hơn, để hoàn thiện tính năng cho Chip các hãng sản xuất theo thời gian đưa ra các Chip có nguồn tài nguyên lớn hơn (RAM, tốc độ Chip, số khuôn dạng Digital, Analog, thư viện tính năng vào khối chức năng).

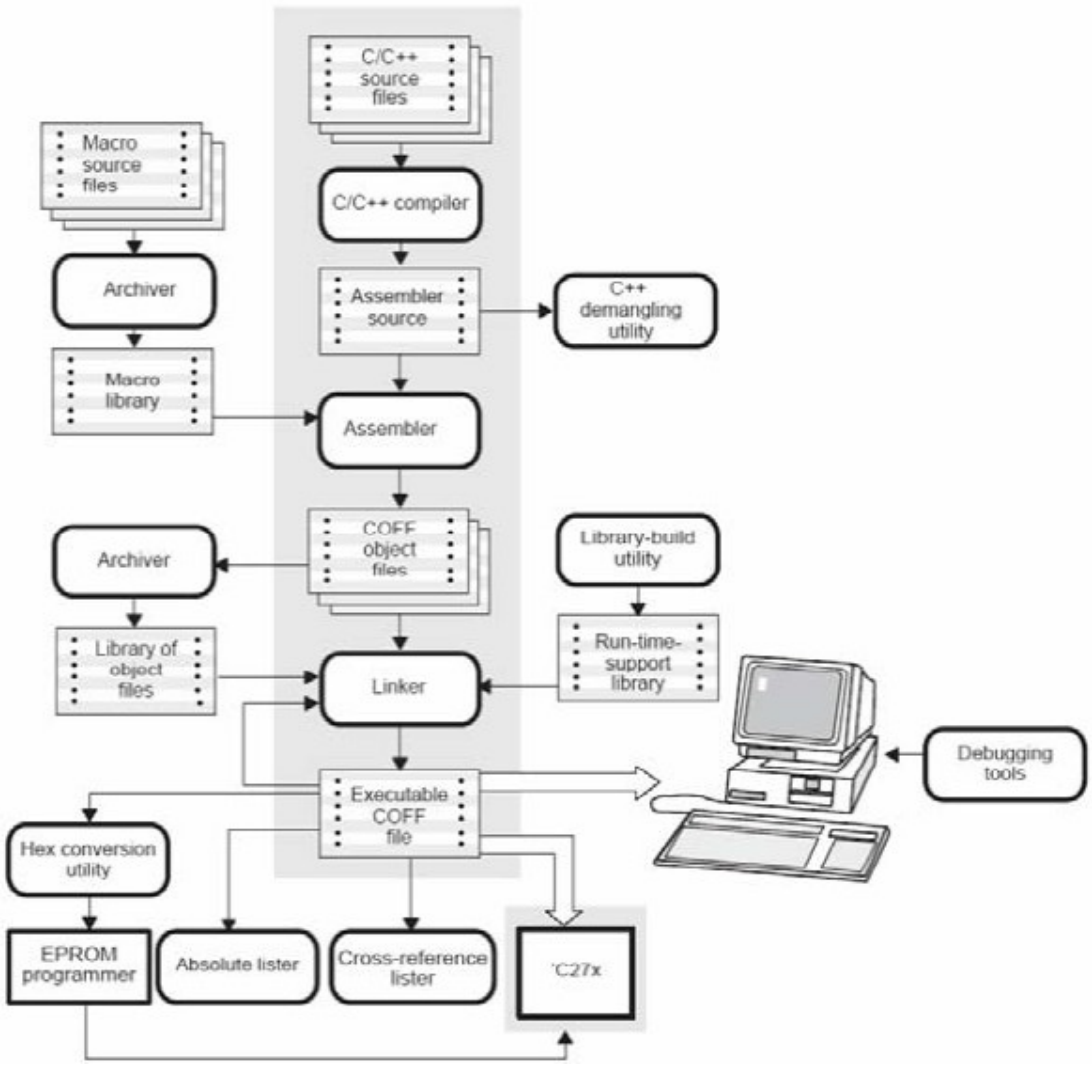
- Thiết kế thêm các Chip có tính năng cần thiết bên ngoài và xây dựng các Module phần mềm liên kết với chúng. Trong trường hợp này sẽ tăng kích thước và giá thành của thiết bị.



MÔI TRƯỜNG PHÁT TRIỂN PHỤC VỤ THIẾT KẾ CHIP PSoC



CÁC BƯỚC TIẾN HÀNH THIẾT KẾ CHIP TRÊN CÔNG NGHỆ PSoC



1. MÔ PHỎNG VÀ GỠ RỐI HỆ THỐNG
2. Hệ thống mô phỏng phần mềm Simulator

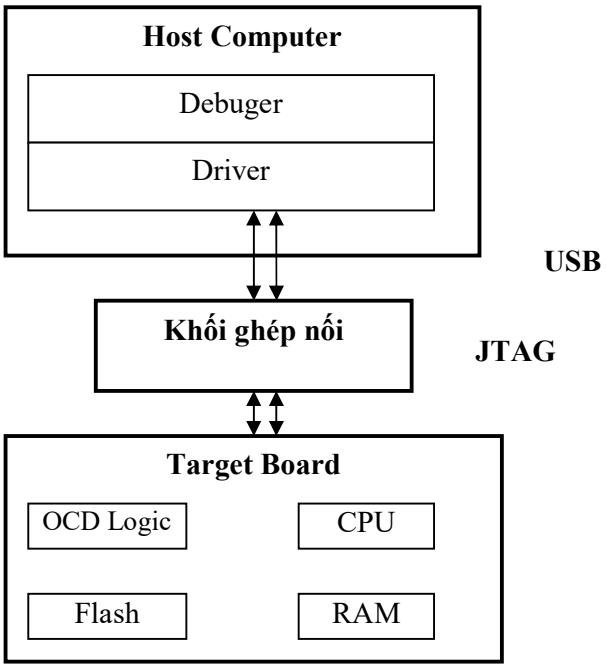
Dùng để mô phỏng sự hotạ động của vi xử lý bằng phần mềm trên máy tính bao gồm các tính năng sau:

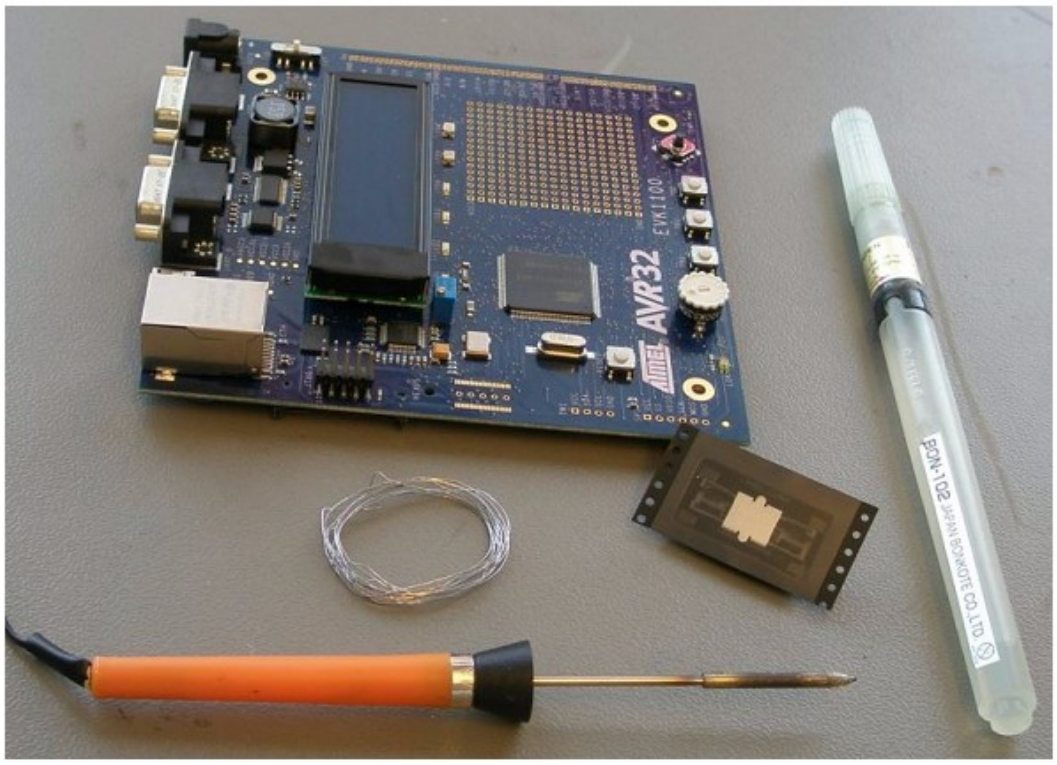
- Mô phỏng sự hoạt động của các lệnh thực hiện chương trình, các lệnh này có thể là ngôn ngữ bâc cao (C) hoặc Assemble.
- Xác định các giá trị dữ liệu trong vùng bộ nhớ (các biến) trong quá trình hoạt động.
- Mô phỏng sự hoạt động của các ngắt.
- Dừng thực hiện chương tình tại các điểm bất kỳ trong chương trình để kiểm tra sự hoạt động của vi xử lý (breakpoint)

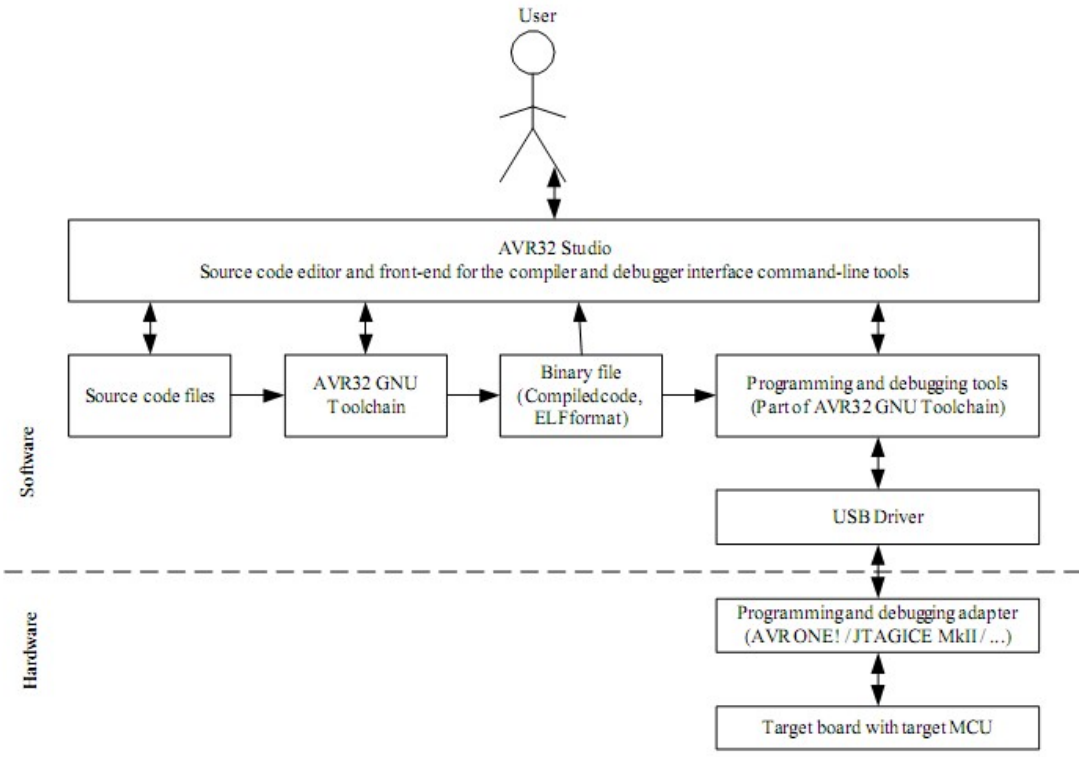
- Mô phỏng sự hoạt động của các cổng vào ra.

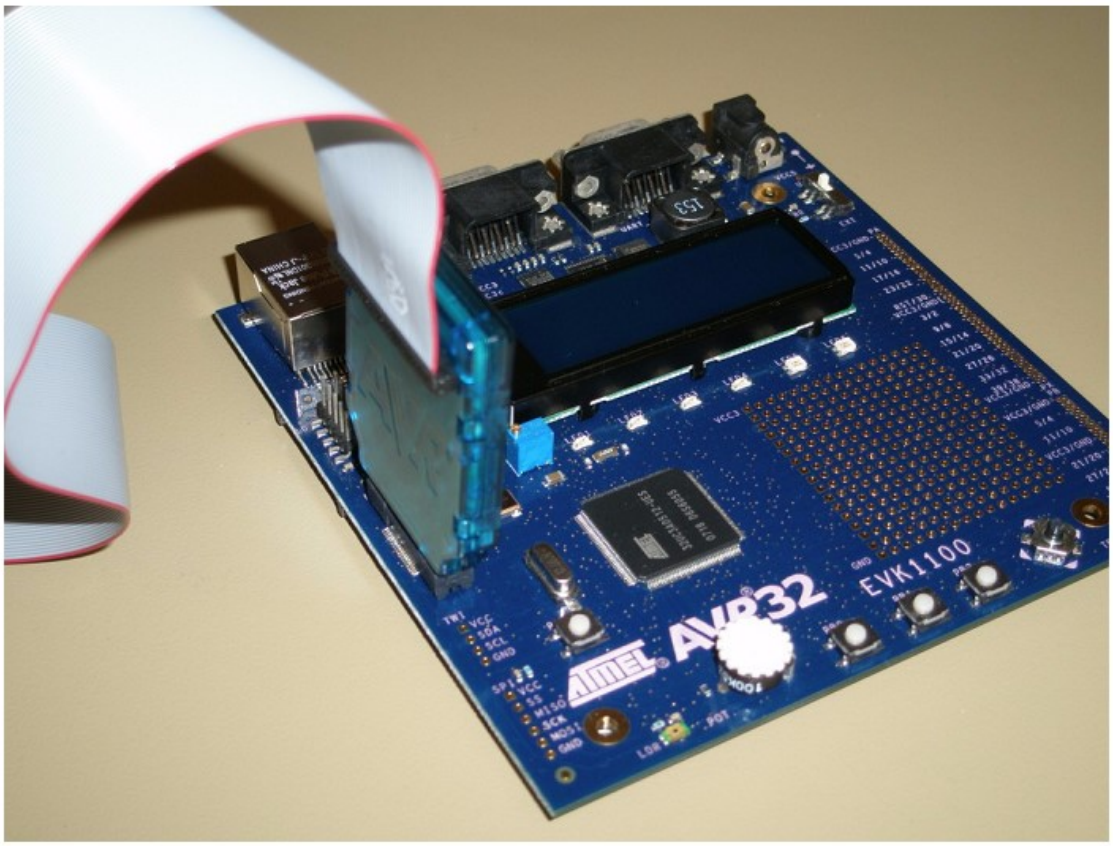
3. Hệ thống mô phỏng phần cứng Emulator.

Dùng để mô phỏng sự hoạt dodọng của vi xử lý và mạch thiết kế dựa trên hệ thống phần mềm trên máy tinh và phần cứng ghép nối đi kèm. Hệ thống Emulator ngoài các khả năng trên còn có thể mô phỏng trên thời gian thực và kiểm tra sự hoạt động của các khối ghép nối bên ngoài như RAM, truyền dữ liệu, giá trị mức điện áp hiện tại ...




AVR32
ATMEL
EVK1100
BON-102
JAPAN BONKOTE CO.,LTD


User
AVR32 Studio
Source code editor and front-end for the compiler and debugger interface command-line tools
Source code files
AVR32 GNU Toolchain
Binary file (Compiled code, ELF format)
Programming and debugging tools (Part of AVR32 GNU Toolchain)
USB Driver
Software
Hardware
Programming and debugging adapter (AVR ONE! / JTAGICE MkII / ...)
Target board with target MCU


AVR32
EVK1100
ATMEL