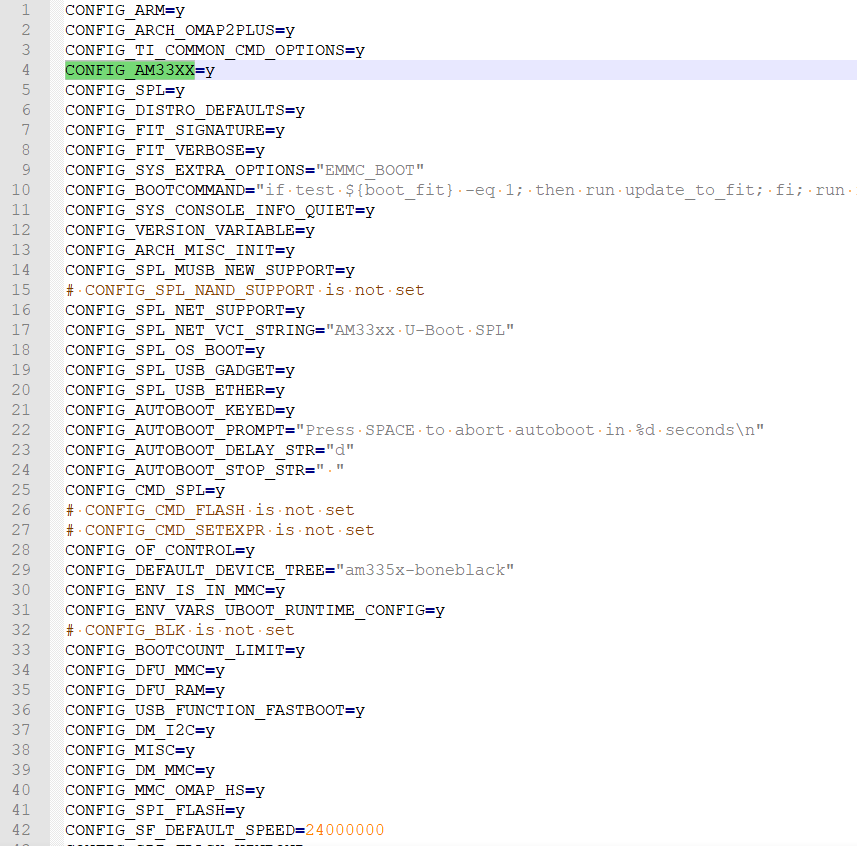
**Top Makefile nhìn thấy tệp .config như thế nào ?**

Cùng nhau xem lại file am335\_boneblack\_vboot\_defconfig.



Định dạng của file này là :

CONFIG\_XXX=value

Có thể thấy rằng giá trị của value này hầu hết là ‘y’. Ngoài ra cũng có các giá trị là chuỗi khác. Đây chính là định dạng cấu hình của phương pháp kbuild. Việc gán cấu hình như vậy sau này sẽ được Makefile sử dụng làm hành vi biên dịch cụ thể. Mở file /driver/timer/Makefile lên ta có ví dụ như sau :

obj-$(CONFIG\_OMAP\_TIMER) += omap-timer.o

Chúng ta có ở đây một biến CONFIG\_OMAP\_TIMER. Ở trong defconfig biến này được gán là y nên kết quả ta thu được là :

obj-y += omap-timer.o

Theo cách cấu hình như thế này ta có thể nối thêm các mục tiêu cho biến “obj-y”. Nếu không có giá trị …. Thì biến này sẽ được gán vào biến “obj-“, được xác định dựa trên tệp cấu hình. Sau này chúng ta có thể biên dịch mục tiêu được chỉ định dựa trên obj-y và mục tiêu được chỉ định bởi obj- có thể được xử lí.

**Làm thế nào để cấu hình đi ra ?**

Bây giờ chúng ta đã hiểu ý nghĩa của việc cấu hình là gì. Vậy câu hỏi đặt ra bây giờ là làm sao top Makefile của chúng ta có thể sử dụng được tệp cấu hình này trong quá trình make? Làm thế nào để file “/driver/timer/Makefile” ở trên nhận ra được biến CONFIG\_OMAP\_TIMER được lưu trữ trong file .config. Bởi vì chúng ta có thể thấy rằng trong “/driver/timer/Makefile không có bất kì dòng lệnh nào giới thiệu về file .config này. Vậy nó được được sử dụng như thế nào nếu như không được include ?

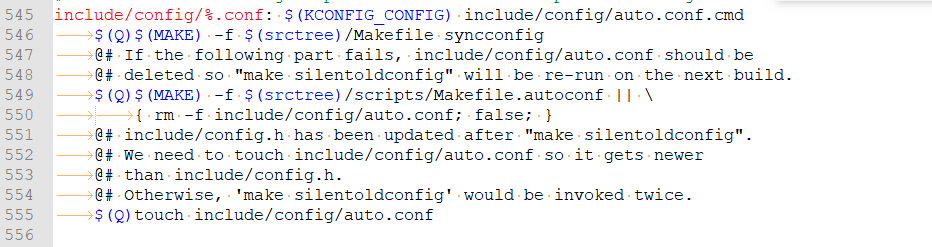
Trước tiên chúng ta tìm trong thư mục gốc Makefile, nơi giới thiệu .config.



Câu lệnh này sẽ gán .config cho KCONFIG\_CONFIG. Lưu ý rằng. Nếu biến KCONFIG\_CONFIG này nếu được được xác định, .config sẽ không ghi đè được lên nó.

Sau đó export biến này để mọi Makefile con đều có thể thấy nó. Tuy nhiên điều này không giúp được gì nhiều vì chúng ta không tìm thấy thứ gì như “include $(KCONFIG\_CONFIG)”.

Chúng ta lại tìm thây đoạn mã sau.



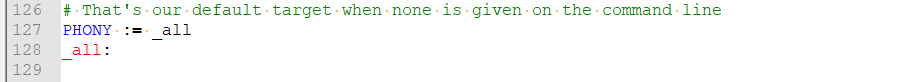
Ở đây “KCONFIG\_CONFIG” là một dependence. Nói cách khác để tạo được mục tiêu thì KCONFIG\_CONFIG phải được xác định tức phải tạo được .config. Nhưng .config này đã được tạo ở các bài viết trước vì vậy ở đây dường như không có bất kì hướng sử dụng nào.

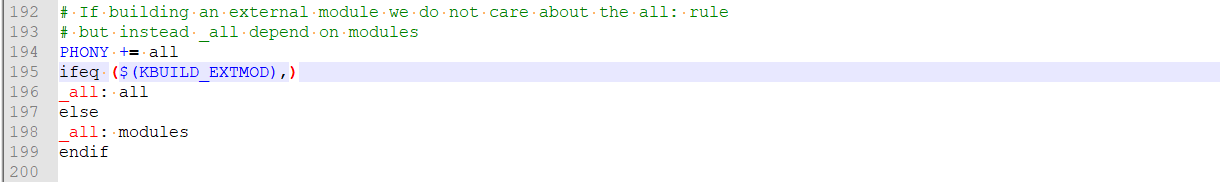
**Liệu chúng ta đang đi sai hướng ?**

Ta tự đưa ra một số câu hỏi, nếu như không có điều gì đề cập đến việc sử dụng cấu hình .config, vậy sau khi gõ lệnh “make” thì uboot.img và uboot.bin được tạo ra như thế nào? Phải chăng trong quá trình gõ make có ẩn chứa việc sử dụng .config này?

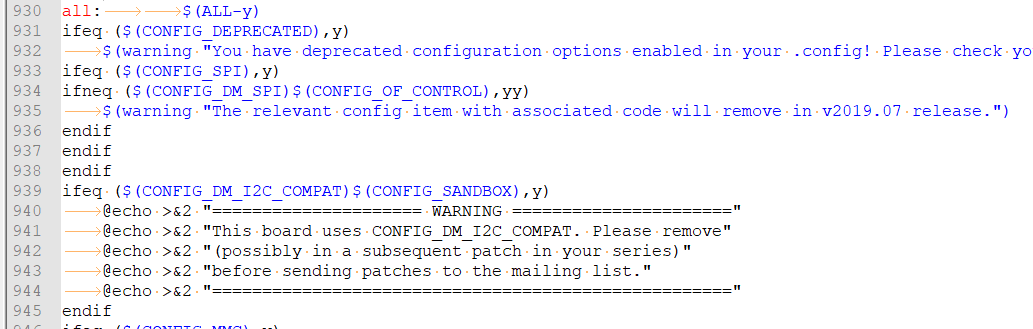
Vì thế chúng ta quyết định tìm hiểu cách thức hoạt động của Makefile khi chúng ta gõ “make” là như thế nào.

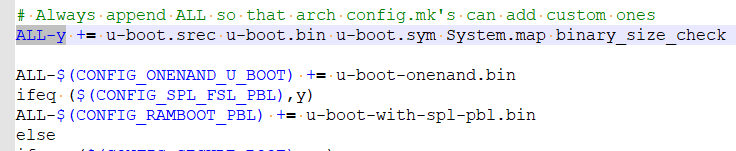
Với những người đã quen với Makefile hẳn đã biết khi gõ “make” thì mục tiêu đầu tiên sẽ được thực hiện và đấy cũng chính là mục tiêu cuối cùng. Tìm trong Makefile ta được đoạn mã như sau :





Theo từng giá trị của KBUILD\_EXTMOD, khi KBUILD\_EXTMOD không được đặt, \_all phụ thuộc vào all, Tìm all ta thấy :

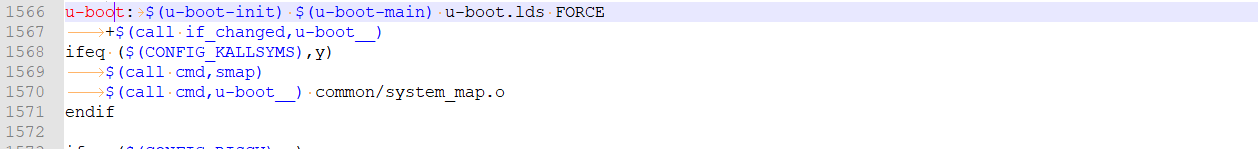
all lại phụ thuộc vào $(ALL-y). Tìm $(ALL-y) :



Đầu tiên ta thấy ALL-y được thêm bởi các giá trị : u-boot.srec u-boot-bin u-boot.sym System.map binary\_size\_check.

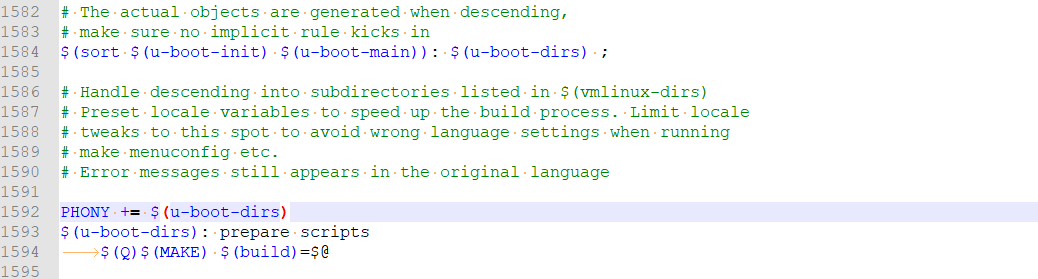
Phần bổ sung tiếp theo phải dựa trên cấu hình, cấu hình ở đây trông giống như định dạng trong file .config? Điều đó có nghĩa là tại thời điểm này nội dung của file .config đã được xác định ?

Ở đây chúng ta sẽ tập trung theo dõi u-boot.bin để tìm xem quá trình xây dựng của nó. Các qui tắc cho u-boot như sau :

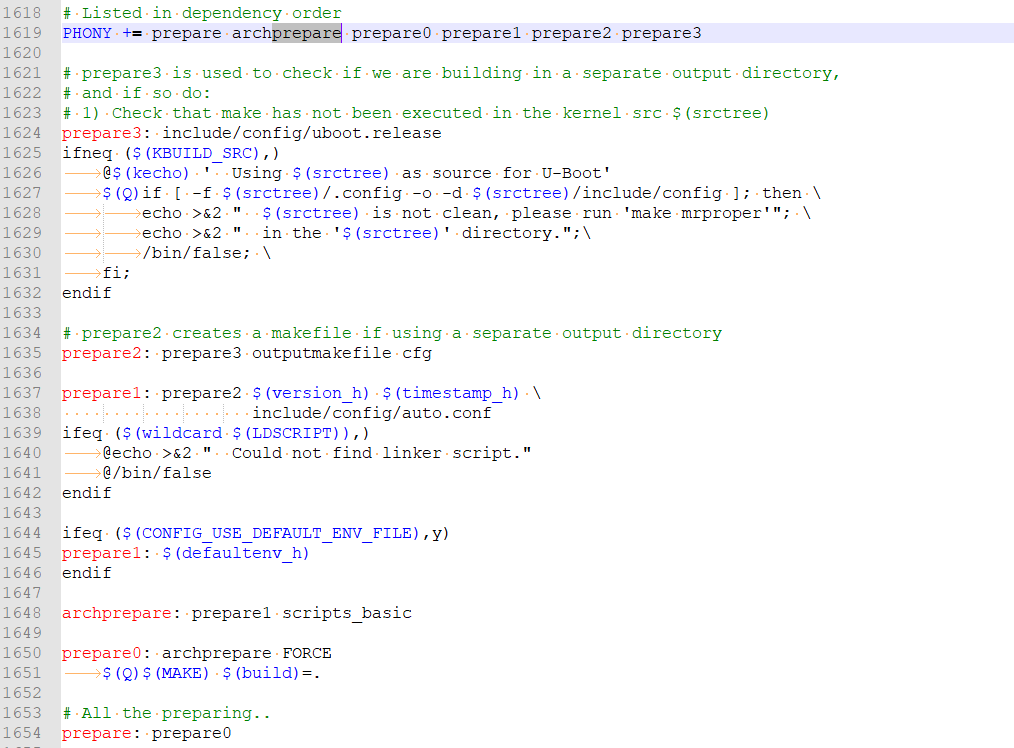


u-boot phụ thuộc vào u-boot-init, u-boot-main và u-boot.lds

Tìm u-boot-init, u-boot-main ta được :



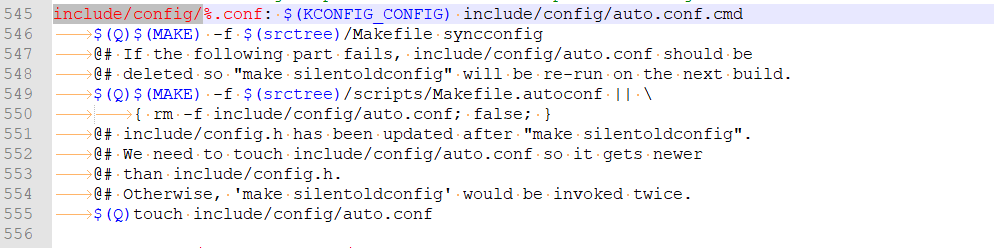
Phụ thuộc của u-boot-init và u-boot-main ở đây là u-boot-dirs. u-boot-dirs lại phụ thuộc vào prepare và scripts. Ta có thể hiểu theo nghĩa đen prepare là sự chuẩn bị cần thiết cho quá trình make của chúng ta.



Chuỗi các phụ thuộc theo thứ tự sau : prepare -> prepare0

-> archprepare -> prepare1 -> prepare2 -> prepare3 -> … -> outputmakefile -> .. -> $(version.h) -> $(timestamp\_h) -> include/config/auto.conf

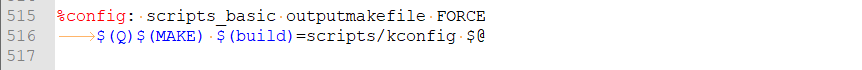
Trở lại qui tắc này ta thấy nó phù hợp với include/config/auto.conf



Action đầu tiên cho qui tắc này sẽ được mở rộng thành

make –f ./Makefile syncconfig

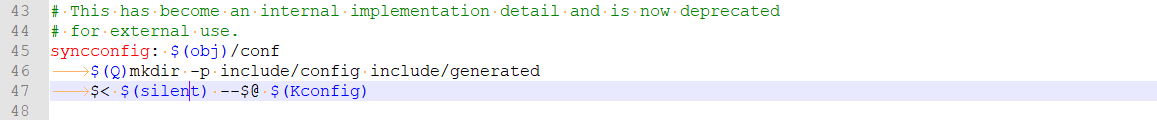
Lệnh này sử dụng top Makefile với tham số là syncconfig. Được mở rộng bởi qui tắc



Mở rộng cuối cùng của nó sẽ là :

$(Q)$(MAKE) –f $(srctree)/scripts/Makefile.build obj=scripts/kconfig syncconfig

Tập lệnh scripts/kconfig chứa qui tắc cho syncconfig :



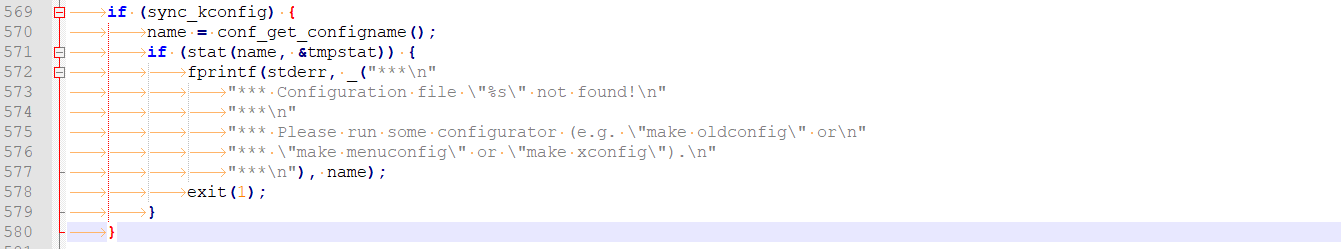
**Mở rộng mã C “conf”**

Phân tích mã nguồn C với đầu vào là syncconfig.



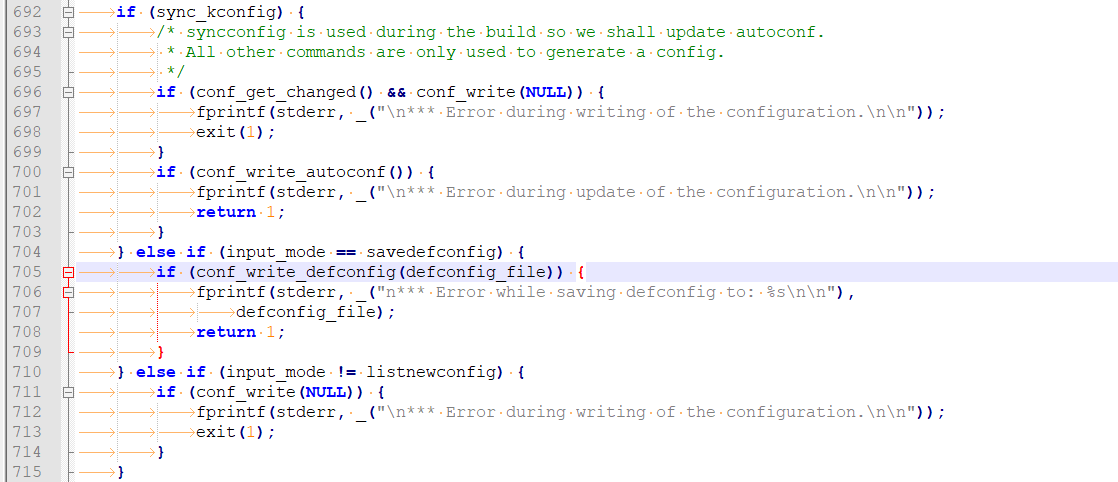
Ở đây cờ sync\_kconfig được đặt thành 1.

Khi đó lệnh if dưới đây được thực thi :



Điều này sẽ gọi tới hàm conf\_get\_configname(), được sử dụng để lấy ra biến môi trường KCONFIG\_CONFIG. Nếu biến môi trường không được tạo thì giá trị mặc định là .config. Nếu .config không tồn tại một thông báo lỗi sẽ được in ra và thoát.

Sau khi lấy được tên file config đoạn mã sau sẽ được thực thi :



Ở đây hàm conf\_write\_autoconf() được sử dụng. Hàm này sẽ tạo tệp include/config/auto.conf, chứa các biến cấu hình được đọc từ .config. Nó giống với .config ngoài trừ các dòng chứa comment sẽ bị xóa – chứa dấu # đầu dòng.