

فهرست بخش های کتاب

عنوان

صفحه

۱	تاریخچه و کاربرد محصولات شرکت ARM	۱
۲	تاریخچه ARM	۱/۱
۲	تاسیس و نامگذاری شرکت ARM	۱/۱/۱
۲	تغییر مالکیت شرکت ARM	۱/۱/۲
۲	نمونه های تجاری	۱/۱/۳
۴	برخی از کاربردها	۱/۱/۴
۴	گواهینامه ها	۱/۱/۵
۵	گواهینامه پردازنده	۱/۱/۵/۱
۵	گواهینامه معماری	۱/۱/۵/۲
۶	گواهینامه پردازنده آماده	۱/۱/۵/۳
۶	میزان فروش و سهم از بازار	۱/۱/۶
۷	اینترنت اشیا	۱/۱/۶/۱
۸	رایانه ها	۱/۱/۶/۲
۸	ابزار رایانه ها	۱/۱/۶/۳
۹	سرورها	۱/۱/۶/۴
۱۰	کاربرد هسته های ARM	۱/۲
۱۰	کاربرد هسته های Cortex-A	۱/۲/۱
۱۲	کاربرد هسته های Cortex-R	۱/۲/۲
۱۳	کاربرد هسته های Cortex-M	۱/۲/۳
۱۵	کاربرد هسته های ایمن	۱/۲/۴
۱۶	کاربرد FPGA با هسته ARM	۱/۲/۵
۱۷	دستگاه ها و تجهیزات ساخته شده از هسته های ARM	۱/۳
۲۱	معماری و مجموعه دستورهای هسته های ARM	۲
۲۲	معماری	۲/۱
۲۲	معماری هاروارد	۲/۱/۱
۲۲	معماری هاروارد	۲/۱/۱/۱
۲۳	معماری اصلاح شده هاروارد	۲/۱/۱/۲
۲۴	معماری وان نیومن	۲/۱/۲
۲۴	معماری وان نیومن	۲/۱/۲/۱
۲۴	مقایسه معماری وان نیومن با هاروارد	۲/۱/۲/۲
۲۵	معماری ARM	۲/۱/۳
۲۶	معماری های AAarch64 و AAarch32 (معماری ۳۲ و ۶۴ بیتی ARM)	۲/۱/۳/۱
۲۶	مجموعه دستورها	۲/۲
۲۶	مجموعه دستورهای CISC	۲/۲/۱
۲۷	مجموعه دستورهای RISC	۲/۲/۲
۲۹	مقایسه مجموعه دستورهای RISC و CISC	۲/۲/۲/۱
۳۰	مجموعه دستورهای معماری ARM	۲/۲/۲/۲
۳۷	ساختار هسته های ARM	۳
۳۸	رابط AMBA	۳/۱
۳۸	ویژگی های رابط AMBA	۳/۱/۱
۳۸	انواع رابط AMBA	۳/۱/۲

۳۹ رابط مرکزی منسجم (CHI) ۳/۱/۲/۱
۴۰ رابط AMBA با ارتباطات گسترش یافته (ACE) ۳/۱/۲/۲
۴۰ رابط پیشرفته توسعه پذیر (AXI) ۳/۱/۲/۳
۴۲ گذرگاه پیشرفته با کارایی بالا (AHB) ۳/۱/۲/۴
۴۳ گذرگاه پیشرفته سیستمی (ASB) ۳/۱/۲/۵
۴۳ گذرگاه پیشرفته اجزای جانبی (APB) ۳/۱/۲/۶
۴۴ گذرگاه پیشرفته برای ردیابی (ATB) ۳/۱/۲/۷
۴۴ خط لوله ۳/۲
۴۵ مرحله های خط لوله در تراشه های RISC ۳/۲/۱
۴۶ واکنشی ۳/۲/۱/۱
۴۶ رمزگشایی ۳/۲/۱/۲
۴۷ اجرا ۳/۲/۱/۳
۴۷ خواندن و نوشتن ۳/۲/۱/۴
۴۷ بازنویسی ۳/۲/۱/۵
۴۷ انواع خطا در خط لوله ۳/۲/۲
۴۸ خطای ساختاری ۳/۲/۲/۱
۴۸ خطای داده ۳/۲/۲/۲
۵۰ خطای کنترلی ۳/۲/۲/۳
۵۱ نمونه های خط لوله ۳/۲/۳
۵۱ خط لوله ARM7TDMI-S ۳/۲/۳/۱
۵۲ خط لوله SC300 ۳/۲/۳/۲
۵۲ خط لوله Cortex-M3 ۳/۲/۳/۳
۵۲ خط لوله Cortex-R4 ۳/۲/۳/۴
۵۳ خط لوله Cortex-A8 ۳/۲/۳/۵
۵۶ خط لوله Cortex-A75 ۳/۲/۳/۶
۵۷ بزرگ/کوچکنویسی ۳/۳
۵۷ بزرگنویسی ۳/۳/۱
۵۷ کوچکنویسی ۳/۳/۲
۵۸ دوگانه نویسی ۳/۳/۳
۵۸ ثبات ها در پردازنده های ARM ۳/۴
۵۸ ثبات های موجود در معماری AArch32 ۳/۴/۱
۶۱ ثبات های موجود در معماری AArch64 ۳/۴/۲
۶۳ حالت ها در پردازنده های ARM ۳/۵
۶۶ استثناها در پردازنده های ARM ۳/۶
۷۰ جدول برداری ۳/۶/۱
۷۰ سطح های استثنا ۳/۶/۲
۷۳ ARM خانواده ۴
۷۵ خانواده کلاسیک (پردازنده های قدیمی) ۴/۱
۷۶ نگارش ARMv4T ۴/۱/۱
۷۷ هسته ARM7TDMI-S ۴/۱/۱/۱
۷۸ نگارش ARMv5TEJ ۴/۱/۲
۸۰ هسته ARM926EJ-S ۴/۱/۲/۱
۸۲ نگارش ARMv6 ۴/۱/۳
۸۲ هسته ARM1176JZ(F)-S ۴/۱/۳/۱
۸۴ خانواده هسته امن ۴/۲
۸۴ نگارش ARMv6-M ۴/۲/۱
۸۴ هسته SC000 ۴/۲/۱/۱
۸۵ نگارش ARMv4T ۴/۲/۲
۸۵ هسته SC100 ۴/۲/۲/۱
۸۶ نگارش ARMv7-M ۴/۲/۳
۸۶ هسته SC300 ۴/۲/۳/۱
۸۷ خانواده Cortex ۴/۳
۸۷ دسته Cortex-M ۴/۳/۱
۸۸ نگارش ARMv6-M ۴/۳/۱/۱
۹۲ نگارش ARMv7-M ۴/۳/۱/۲

۹۶	ARMv8-M نگارش ۴/۳/۱/۳
۹۹	Cortex-R دسته ۴/۳/۲
۱۰۲	ARMv7-R نگارش ۴/۳/۲/۱
۱۰۹	ARMv8-R نگارش ۴/۳/۲/۲
۱۱۲	Cortex-A دسته ۴/۳/۳
۱۱۴	ARMv7-A نگارش ۴/۳/۳/۱ (۳۲-بیتی)
۱۲۶	ARMv8-A نگارش ۴/۳/۳/۲ (۶۴-بیتی)
۱۴۷	ARM نگارش‌های
۱۵۱	۴/۵ پردازنده‌های تولیدشده توسط دیگر شرکت‌ها
۱۵۳	۵ آشنایی با نرم‌افزار کیل
۱۵۴	۵/۱ چگونگی دریافت و نصب نرم‌افزار کیل و بسته‌های آن
۱۵۴	۵/۱/۱ چگونگی دریافت نرم‌افزار کیل
۱۵۶	۵/۱/۲ چگونگی نصب نرم‌افزار کیل
۱۵۷	۵/۱/۳ چگونگی کرک نرم‌افزار کیل
۱۵۹	۵/۱/۴ چگونگی اجرای نرم‌افزار کیل
۱۵۹	۵/۱/۵ چگونگی دریافت، نصب و بروزرسانی بسته‌ها در نرم‌افزار کیل
۱۶۰	۵/۲ معرفی بخش‌های مختلف نرم‌افزار کیل
۱۶۰	۵/۲/۱ معرفی بخش‌های کلی نرم‌افزار کیل
۱۶۱	۵/۲/۲ توضیحاتی در مورد منوی نرم‌افزار کیل
۱۶۱	۵/۲/۳ توضیحاتی در مورد نوار ابزارهای نرم‌افزار کیل
۱۶۳	۵/۳ ایجاد پروژه و تنظیمات آن در نرم‌افزار کیل
۱۶۳	۵/۳/۱ چگونگی ایجاد پروژه
۱۶۸	۵/۳/۲ بازکردن یک پروژه از قبل ایجادشده
۱۶۸	۵/۳/۳ شبیه‌سازی برنامه در نرم‌افزار کیل
۱۶۹	۵/۴ چگونگی ترجمه (Compile) پروژه و انتقال آن به برد مدار چاپی
۱۶۹	۵/۴/۱ چگونگی ترجمه پروژه
۱۷۰	۵/۴/۲ انتقال پروژه به برد مدار چاپی
۱۷۲	۶ آشنایی با پردازنده Cortex-M4
۱۷۳	۶/۱ توضیحاتی در مورد پردازنده Cortex-M4
۱۷۳	۶/۲ ویژگی‌های تراشه‌های LPC408x/7x
۱۷۶	۶/۳ خلاصه مشخصات تراشه‌های سری LPC408x/7x
۱۷۶	۶/۴ مروری کلی بر معماری تراشه‌های سری LPC408x/7x
۱۷۶	۶/۵ حافظه فلش داخلی
۱۷۷	۶/۶ حافظه SRAM داخلی
۱۷۸	۶/۷ نمودار کلی با جزئیات
۱۷۹	۷ کنترل تراشه و ضربان‌دهی
۱۸۰	۷/۱ خلاصه‌ای از کاربردهای ضربان‌دهی و کنترل توان
۱۸۱	۷/۲ توضیحات مربوط به ثبات‌های بخش کنترل تراشه و ضربان‌دهی
۱۸۲	۷/۲/۱ ثبات‌های PLL
۱۸۳	۷/۲/۲ ثبات‌های کنترل توان
۱۸۶	۷/۲/۳ ثبات‌های انتخاب ضربان و تقسیم‌کننده ضربان
۱۸۹	۷/۲/۴ وقفه‌های خارجی
۱۹۱	۷/۲/۵ راه‌اندازی مجدد تراشه و اجزای جانبی
۱۹۲	۷/۲/۶ کنترل و تنظیم تاخیر EMC
۱۹۴	۷/۲/۷ ثبات‌های مختلف کنترل تراشه
۱۹۷	۷/۳ راه‌اندازی مجدد تراشه
۱۹۹	۷/۴ کنترل راه‌اندازی مجدد اجزای جانبی
۱۹۹	۷/۵ شناسایی افت تغذیه
۱۹۹	۷/۶ نوسان‌گرها
۱۹۹	۷/۶/۱ نوسان‌گر RC درونی
۱۹۹	۷/۶/۲ نوسان‌گر اصلی
۲۰۰	۷/۶/۲/۱ شروع بکار نوسان‌گر اصلی
۲۰۱	۷/۶/۳ نوسان‌گر RTC
۲۰۱	۷/۶/۴ نوسان‌گر نگهبان
۲۰۱	۷/۷ انتخابگرهای منبع ضربان

۲۰۱ PLL0 و PLL1 (حلقه قفل فاز ۰ و ۱)
۲۰۲ ۷/۸/۱ توضیحاتی در مورد ثبات‌های PLL
۲۰۳ ۷/۸/۲ PLLها و حالت کاهش مصرف توان
۲۰۳ ۷/۸/۳ محاسبه بسامد PLL
۲۰۳ ۷/۸/۴ تعیین تنظیمات PLL
۲۰۵ ۷/۸/۵ بخش پیکربندی PLL
۲۰۵ ۷/۸/۵/۱ تغییر منبع ضربان از PLL به منبع دیگر
۲۰۵ ۷/۸/۶ مثال‌هایی از پیکربندی PLL
۲۰۶ ۷/۹ کنترل توان
۲۰۷ ۷/۹/۱ حالت خواب
۲۰۷ ۷/۹/۲ حالت خواب عمیق
۲۰۸ ۷/۹/۳ حالت کاهش مصرف توان
۲۰۸ ۷/۹/۴ حالت کاهش مصرف توان عمیق
۲۰۸ ۷/۹/۵ بیدار شدن از حالت‌های کاهش مصرف توان
۲۰۹ ۷/۹/۶ نکاتی در مورد استفاده از کنترل توان
۲۰۹ ۷/۹/۷ منابع توان
۲۰۹ ۷/۱۰ زمان‌سنج بیداری
۲۱۱ ۸ حافظه‌های داخلی و نحوه کنترل آنها
۲۱۲ ۸/۱ انواع حافظه‌ها
۲۱۲ ۸/۱/۱ حافظه‌هایی از نوع ROM
۲۱۲ ۸/۱/۱/۱ حافظه ROM
۲۱۲ ۸/۱/۱/۲ حافظه PROM
۲۱۲ ۸/۱/۱/۳ حافظه EPROM
۲۱۳ ۸/۱/۱/۴ حافظه EEPROM
۲۲۰ ۸/۱/۱/۵ حافظه فلش
۲۲۶ ۸/۱/۲ حافظه‌هایی از نوع RAM
۲۲۶ ۸/۱/۲/۱ حافظه NVRAM
۲۲۶ ۸/۱/۲/۲ حافظه SRAM
۲۲۸ ۸/۱/۲/۳ حافظه DRAM
۲۳۲ ۸/۲ فضای آدرس و نگاشت حافظه
۲۳۲ ۸/۲/۱ نگاشت حافظه و آدرس‌دهی اجزای جانبی
۲۳۴ ۸/۲/۲ اجزای داخلی
۲۳۴ ۸/۲/۲/۱ آدرس اجزای متصل به گذرگاه AHB
۲۳۴ ۸/۲/۲/۲ آدرس اجزای متصل به گذرگاه APB
۲۳۵ ۸/۲/۳ نگاشت دوباره حافظه
۲۳۶ ۸/۲/۴ اولویت‌بندی توسط AHB
۲۳۶ ۸/۳ شتاب‌دهنده فلش
۲۳۶ ۸/۳/۱ واحد شتاب‌دهنده فلش
۲۳۷ ۸/۳/۱/۱ بانک حافظه فلش
۲۳۷ ۸/۳/۱/۲ برنامه‌ریزی فلش
۲۳۸ ۸/۳/۲ ثبات پیکربندی شتاب‌دهنده حافظه فلش (FLASHCFG)
۲۳۸ ۸/۳/۳ عملیات خواندن از حافظه موقت
۲۴۱ ۹ حافظه‌های SDRAM
۲۴۲ ۹/۱ حافظه "SDR SDRAM"
۲۴۳ ۹/۱/۱ زمان‌بندی در "SDR SDRAM"
۲۴۳ ۹/۱/۲ موج‌های کنترل در "SDR SDRAM"
۲۴۴ ۹/۱/۲/۱ موج‌های فرمان
۲۴۴ ۹/۱/۲/۲ پایه‌های انتخاب بانک (BA _n)
۲۴۴ ۹/۱/۲/۳ پایه‌های آدرس‌دهی (A[n:0])
۲۴۴ ۹/۱/۳ فرمان‌ها در حافظه SDR
۲۴۵ ۹/۲ خانواده حافظه‌های "DDR SDRAM"
۲۴۶ ۹/۲/۱ دسته حافظه‌های DDR
۲۴۸ ۹/۲/۱/۱ حافظه DDR نسل اول (DDR)
۲۵۲ ۹/۲/۱/۲ حافظه DDR نسل دوم (DDR2)
۲۵۷ ۹/۲/۱/۳ حافظه DDR نسل سوم (DDR3)

۲۶۳ حافظه DDR نسل چهارم (DDR4)
۲۷۰ حافظه DDR نسل پنجم (DDR5)
۲۷۰ دسته حافظه‌های GDDR
۲۷۲ حافظه GDDR نسل اول (GDDR)
۲۷۶ حافظه GDDR نسل دوم (GDDR2)
۲۸۱ حافظه GDDR نسل سوم (GDDR3)
۲۸۷ حافظه GDDR نسل چهارم (GDDR4)
۲۹۳ حافظه GDDR نسل پنجم (GDDR5)
۳۰۳ حافظه GDDR نسل ششم (GDDR5X)
۳۱۲ حافظه GDDR نسل هفتم (GDDR6)
۳۱۳ دسته حافظه‌های LPDDR
۳۱۴ حافظه LPDDR نسل اول (LPDDR)
۳۱۸ حافظه LPDDR نسل دوم (LPDDR2)
۳۲۳ حافظه LPDDR نسل سوم (LPDDR3)
۳۲۹ حافظه LPDDR نسل چهارم (LPDDR4)
۳۳۶ حافظه LPDDR نسل پنجم (LPDDR4X)
۳۳۷	۱۰ کنترل‌کننده حافظه خارجی (EMC)
۳۳۸ ۱۰/۱ پیکربندی EMC
۳۳۹ ۱۰/۲ ویژگی‌های EMC
۳۳۹ ۱۰/۳ توضیحاتی در مورد کاربردهای EMC
۳۴۱ ۱۰/۴ کارکرد با مصرف توان پایین
۳۴۲ ۱۰/۴/۱ ورود SDRAM به حالت خواب عمیق
۳۴۲ ۱۰/۴/۲ تازمسازی بخشی از SDRAM
۳۴۲ ۱۰/۵ انتخاب تراشه‌های حافظه
۳۴۲ ۱۰/۶ راه‌اندازی مجدد EMC
۳۴۳ ۱۰/۷ حالت جابجایی آدرس
۳۴۳ ۱۰/۸ نگاشت حافظه و غیرفعال‌سازی جابجایی پشت‌سرهم داده
۳۴۳ ۱۰/۹ وصل‌کردن SDRAM به EMC
۳۴۳ ۱۰/۹/۱ تنظیم ثبات حالت
۳۴۴ ۱۰/۱۰ توضیحاتی در مورد پایه‌های EMC
۳۴۵ ۱۰/۱۱ توضیحاتی در مورد ثبات‌های EMC
۳۶۰ ۱۰/۱۲ رابط حافظه ثابت خارجی
۳۶۰ ۱۰/۱۲/۱ پیکربندی‌های گذرگاهی با پهنای ۳۲-بیتی
۳۶۱ ۱۰/۱۲/۲ پیکربندی‌های گذرگاهی با پهنای ۱۶-بیتی
۳۶۱ ۱۰/۱۲/۳ پیکربندی گذرگاهی با پهنای ۸-بیتی
۳۶۱ ۱۰/۱۲/۴ نمونه‌ای از گذرگاهی با پهنای ۳۲-بیتی
۳۶۳	۱۱ کنترل‌کننده وقفه برداری تودرتو
۳۶۴ ۱۱/۱ ویژگی‌های NVIC
۳۶۴ ۱۱/۲ منابع وقفه
۳۶۶ ۱۱/۳ نگاشت مجدد جدول برداری
۳۶۶ ۱۱/۴ توضیحاتی در مورد ثبات‌های NVIC
۳۷۷	۱۲ پایه‌ها و پیکربندی آنها
۳۷۸ ۱۲/۱ چینش پایه‌ها
۳۹۲ ۱۲/۲ پیکربندی پایه‌های ورودی/خروجی
۳۹۳ ۱۲/۲/۱ ثبات‌های IOCON
۳۹۴ ۱۲/۲/۱/۱ کاربرد پایه‌ها
۳۹۴ ۱۲/۲/۱/۲ حالت پایه
۳۹۴ ۱۲/۲/۱/۳ پسماند
۳۹۴ ۱۲/۲/۱/۴ معکوس‌کردن ورودی
۳۹۵ ۱۲/۲/۱/۵ حالت آنالوگ/دیجیتال
۳۹۵ ۱۲/۲/۱/۶ فیلتر ورودی
۳۹۵ ۱۲/۲/۱/۷ نرخ تأخیر خروجی
۳۹۵ ۱۲/۲/۱/۸ حالت‌های I2C
۳۹۵ ۱۲/۲/۱/۹ حالت درین باز
۳۹۵ ۱۲/۲/۱/۱۰ فعال‌سازی DAC

۳۹۶	IOCON توضیح ثبات‌های	۱۲/۲/۲
۳۹۹	محتویات ثبات پیکر مبنی I/O (IOCON)	۱۲/۲/۲/۱
۴۰۷	پایه‌های ورودی/خروجی همه‌منظوره (GPIO)	۱۲/۳
۴۰۷	پیکر مبنی پایه‌های GPIO	۱۲/۳/۱
۴۰۷	ویژگی‌ها	۱۲/۳/۲
۴۰۷	درگاه‌های ورودی/خروجی دیجیتال	۱۲/۳/۲/۱
۴۰۷	ایجاد وقفه پایه‌های دیجیتال	۱۲/۳/۲/۲
۴۰۷	کاربردهای GPIO	۱۲/۳/۳
۴۰۷	توضیح پایه‌های GPIO	۱۲/۳/۴
۴۰۸	توضیح ثبات‌های پیکر مبنی GPIO	۱۲/۳/۵
۴۰۹	ثبات‌های درگاه GPIO	۱۲/۳/۵/۱
۴۱۱	ثبات‌های وقفه GPIO	۱۲/۳/۵/۲
۴۱۹	نکاتی در مورد استفاده از GPIO	۱۲/۳/۶
۴۱۹	نشان دادن مقداری در یک درگاه GPIO	۱۲/۳/۶/۱
۴۱۹	مقایسه نوشتن در ثبات‌های FIONSET/FIONCLR با ثبات FIONPIN	۱۲/۳/۶/۲
۴۲۰	رابط فلش SPI چهارگانه (SPIFI)	
۴۲۱	پیکر مبنی رابط SPIFI	۱۳/۱
۴۲۱	ویژگی‌های رابط SPIFI	۱۳/۲
۴۲۱	توضیحاتی در مورد پایه‌های رابط SPIFI	۱۳/۳
۴۲۲	حافظه‌های حمایت شده توسط رابط SPIFI	۱۳/۴
۴۲۳	سخت‌افزار رابط SPIFI	۱۳/۵
۴۲۳	ثبات‌های رابط SPIFI	۱۳/۶
۴۲۷	توضیحات کاربردی	۱۳/۷
۴۲۷	ارسال داده	۱۳/۷/۱
۴۲۹	نیازها و قابلیت‌های نرم‌افزاری	۱۳/۷/۲
۴۳۰	کارکرد DMA در حالت جانبی	۱۳/۷/۳
۴۳۱	رابط کارت‌های SD	
۴۳۲	پیکر مبنی	۱۴/۱
۴۳۲	توضیحاتی در مورد پایه‌های رابط SD	۱۴/۲
۴۳۲	کاربردهای رابط SD	۱۴/۳
۴۳۲	کارت SD	۱۴/۳/۱
۴۳۲	پایه‌های گذرگاه کارت SD	۱۴/۳/۱/۱
۴۳۳	کارت MMC	۱۴/۳/۲
۴۳۳	توضیحاتی در مورد رابط SD	۱۴/۳/۳
۴۳۴	بخش سازگارکننده	۱۴/۳/۳/۱
۴۳۴	واحد کنترل	۱۴/۳/۳/۲
۴۳۴	خط فرمان	۱۴/۳/۳/۳
۴۳۴	چرخه وضعیت خط فرمان	۱۴/۳/۳/۴
۴۳۵	ساختار فرمان	۱۴/۳/۳/۵
۴۳۶	خط داده	۱۴/۳/۳/۶
۴۳۶	چرخه وضعیت خط داده	۱۴/۳/۳/۷
۴۳۸	شمارنده داده	۱۴/۳/۳/۸
۴۳۸	حالت گذرگاه	۱۴/۳/۳/۹
۴۳۹	وضعیت CRC	۱۴/۳/۳/۱۰
۴۳۹	پرچم‌های وضعیت	۱۴/۳/۳/۱۱
۴۳۹	CRC داده	۱۴/۳/۳/۱۲
۴۳۹	FIFO داده	۱۴/۳/۳/۱۳
۴۴۱	رابط APB	۱۴/۳/۳/۱۴
۴۴۱	بخش وقفه	۱۴/۳/۳/۱۵
۴۴۱	ثبات‌های رابط SD	۱۴/۴
۴۴۹	گذرگاه I2C	
۴۵۰	پیکر مبنی گذرگاه I2C	۱۵/۱
۴۵۰	ویژگی‌های گذرگاه I2C	۱۵/۲
۴۵۰	کاربردهای گذرگاه I2C	۱۵/۳
۴۵۰	توضیحاتی در مورد گذرگاه I2C	۱۵/۴

۴۵۱	۱۵/۴/۱	حالت بسیار سریع I2C
۴۵۱	۱۵/۵	توضیح پایه‌های گذرگاه I2C
۴۵۲	۱۵/۶	حالت‌های اجرایی بخش I2C
۴۵۲	۱۵/۶/۱	حالت فرستنده اصلی
۴۵۳	۱۵/۶/۲	حالت گیرنده اصلی
۴۵۴	۱۵/۶/۳	حالت گیرنده فرعی
۴۵۴	۱۵/۶/۴	حالت فرستنده فرعی
۴۵۵	۱۵/۷	عملکرد و اجرای بخش I2C
۴۵۵	۱۵/۷/۱	ثبات‌های آدرس I2CnADR0 تا I2CnADR3
۴۵۵	۱۵/۷/۲	فیلترهای ورودی
۴۵۵	۱۵/۷/۳	ثبات‌های پوشاننده آدرس I2CnMASK0 تا I2CnMASK3
۴۵۶	۱۵/۷/۴	مقایسه‌گرها
۴۵۶	۱۵/۷/۵	ثبات مبدل جابجایی I2CnDAT
۴۵۶	۱۵/۷/۶	اولویت‌بندی و همزمان‌سازی
۴۵۷	۱۵/۷/۷	تولیدکننده ضربان
۴۵۷	۱۵/۷/۸	زمان‌بندی و کنترل
۴۵۷	۱۵/۷/۹	ثبات‌های کنترل (I2CnCON)، I2CnCONSET و I2CnCONCLR
۴۵۷	۱۵/۷/۱۰	ثبات وضعیت (I2CnSTAT)
۴۵۸	۱۵/۸	توضیحاتی در مورد ثبات‌های بخش I2C
۴۶۴	۱۵/۹	حالت‌های اجرایی در بخش I2C
۴۶۴	۱۵/۹/۱	حالت فرستنده اصلی
۴۶۵	۱۵/۹/۲	حالت گیرنده اصلی
۴۶۶	۱۵/۹/۳	حالت گیرنده فرعی
۴۶۷	۱۵/۹/۴	حالت فرستنده فرعی
۴۶۸	۱۵/۹/۵	جدول کدهای وضعیت با جزئیات آن‌ها
۴۷۱	۱۵/۹/۶	وضعیت‌های متفرقه
۴۷۲	۱۵/۹/۷	موارد ویژه
۴۷۲	۱۵/۹/۷/۱	ارسال بیت شروع مجدد به صورت همزمان از دو تراشه اصلی
۴۷۲	۱۵/۹/۷/۲	جابجایی داده بعد از، از دست رفتن اولویت‌بندی
۴۷۲	۱۵/۹/۷/۳	اصرار در دسترسی به گذرگاه I2C
۴۷۲	۱۵/۹/۷/۴	دچار مشکل شدن گذرگاه I2C بوسیله شدن خط SCL یا SDA
۴۷۳	۱۵/۹/۷/۵	خطای گذرگاه
۴۷۳	۱۵/۹/۸	روال‌های سرویس وضعیت بخش I2C
۴۷۴	۱۵/۹/۸/۱	مقداردهی اولیه
۴۷۴	۱۵/۹/۸/۲	سرویس وقفه I2C
۴۷۴	۱۵/۹/۸/۳	روال‌های سرویس وضعیت
۴۷۴	۱۵/۹/۸/۴	هماهنگ کردن سرویس‌های وضعیت با برنامه
۴۷۴	۱۵/۱۰	مثال‌های نرم‌افزاری
۴۷۴	۱۵/۱۰/۱	روال مقداردهی اولیه
۴۷۵	۱۵/۱۰/۲	شروع حالت فرستنده اصلی
۴۷۵	۱۵/۱۰/۳	شروع حالت گیرنده اصلی
۴۷۵	۱۵/۱۰/۴	روال وقفه I2C
۴۷۵	۱۵/۱۰/۵	وضعیت‌های غیروابسته به چهار حالت اجرایی I2C
۴۷۵	۱۵/۱۰/۵/۱	وضعیت 0x00
۴۷۵	۱۵/۱۰/۵/۲	وضعیت‌های حالت اصلی
۴۷۵	۱۵/۱۰/۵/۳	وضعیت 0x08
۴۷۶	۱۵/۱۰/۵/۴	وضعیت 0x10
۴۷۶	۱۵/۱۰/۶	وضعیت‌های حالت فرستنده اصلی
۴۷۶	۱۵/۱۰/۶/۱	وضعیت 0x18
۴۷۶	۱۵/۱۰/۶/۲	وضعیت 0x20
۴۷۶	۱۵/۱۰/۶/۳	وضعیت 0x28
۴۷۷	۱۵/۱۰/۶/۴	وضعیت 0x30
۴۷۷	۱۵/۱۰/۶/۵	وضعیت 0x38
۴۷۷	۱۵/۱۰/۷	وضعیت‌های گیرنده اصلی
۴۷۷	۱۵/۱۰/۷/۱	وضعیت 0x40
۴۷۷	۱۵/۱۰/۷/۲	وضعیت 0x48

۴۷۷ 0x50 وضعیت ۱۵/۱۰/۷/۳
۴۷۷ 0x58 وضعیت ۱۵/۱۰/۷/۴
۴۷۸ ۱۵/۱۰/۸/۸ وضعیت‌های گیرنده فرعی
۴۷۸ 0x60 وضعیت ۱۵/۱۰/۸/۱
۴۷۸ 0x68 وضعیت ۱۵/۱۰/۸/۲
۴۷۸ 0x70 وضعیت ۱۵/۱۰/۸/۳
۴۷۸ 0x78 وضعیت ۱۵/۱۰/۸/۴
۴۷۹ 0x80 وضعیت ۱۵/۱۰/۸/۵
۴۷۹ 0x88 وضعیت ۱۵/۱۰/۸/۶
۴۷۹ 0x90 وضعیت ۱۵/۱۰/۸/۷
۴۷۹ 0x98 وضعیت ۱۵/۱۰/۸/۸
۴۷۹ 0xA0 وضعیت ۱۵/۱۰/۸/۹
۴۸۰ ۱۵/۱۰/۹/۹ وضعیت‌های فرستنده فرعی
۴۸۰ 0xA8 وضعیت ۱۵/۱۰/۹/۱
۴۸۰ 0xB0 وضعیت ۱۵/۱۰/۹/۲
۴۸۰ 0xB8 وضعیت ۱۵/۱۰/۹/۳
۴۸۰ 0xC0 وضعیت ۱۵/۱۰/۹/۴
۴۸۰ 0xC8 وضعیت ۱۵/۱۰/۹/۵
۴۸۱ ۱۶ درگاه‌های سریال ۰ تا ۴ (UART[4:0])
۴۸۲ ۱۶/۱ درگاه‌های سریال ۳/۲/۰ (UART0,2,3)
۴۸۲ ۱۶/۱/۱ پیکر مبنی درگاه‌های سریال ۳/۲/۰
۴۸۲ ۱۶/۱/۲ ویژگی‌های درگاه‌های سریال ۳/۲/۰
۴۸۲ ۱۶/۱/۳ معماری درگاه‌های سریال ۳/۲/۰
۴۸۳ ۱۶/۱/۴ توضیح پایه‌های درگاه‌های سریال ۳/۲/۰
۴۸۴ ۱۶/۱/۵ توضیح ثبات‌های درگاه‌های سریال ۳/۲/۰
۴۹۷ ۱۶/۲ درگاه سریال ۱ (UART1)
۴۹۷ ۱۶/۲/۱ پیکر مبنی درگاه سریال ۱
۴۹۸ ۱۶/۲/۲ ویژگی‌های درگاه سریال ۱
۴۹۸ ۱۶/۲/۳ معماری درگاه سریال ۱
۴۹۹ ۱۶/۲/۴ توضیح پایه‌های درگاه سریال ۱
۵۰۰ ۱۶/۲/۵ توضیح ثبات‌های درگاه سریال ۱
۵۱۴ ۱۶/۳ درگاه سریال ۴ (UART4)
۵۱۵ ۱۶/۳/۱ پیکر مبنی درگاه سریال ۴
۵۱۵ ۱۶/۳/۲ ویژگی‌های درگاه سریال ۴
۵۱۵ ۱۶/۳/۳ معماری درگاه سریال ۴
۵۱۶ ۱۶/۳/۴ توضیح پایه‌های درگاه سریال ۴
۵۱۶ ۱۶/۳/۵ توضیح ثبات‌های درگاه سریال ۴
۵۳۳ ۱۷ زمان‌سنج‌ها
۵۳۴ ۱۷/۱ زمان‌سنج‌های ۰ تا ۳
۵۳۴ ۱۷/۱/۱ پیکر مبنی زمان‌سنج‌های ۰ تا ۳
۵۳۴ ۱۷/۱/۲ ویژگی‌های زمان‌سنج‌های ۰ تا ۳
۵۳۴ ۱۷/۱/۳ کاربردهای زمان‌سنج‌های ۰ تا ۳
۵۳۴ ۱۷/۱/۴ توضیحاتی در مورد زمان‌سنج‌های ۰ تا ۳
۵۳۵ ۱۷/۱/۵ توضیحاتی در مورد پایه‌های زمان‌سنج/شمارنده ۰ تا ۳
۵۳۶ ۱۷/۱/۵/۱ پایه‌های چندگانه برای ثبات‌های برابری و انداز مگیری
۵۳۶ ۱۷/۱/۶ توضیح ثبات‌های زمان‌سنج‌های ۰ تا ۳
۵۴۱ ۱۷/۱/۷ مثالی در مورد کارکرد زمان‌سنج
۵۴۲ ۱۷/۲ زمان‌سنج ثابتهای
۵۴۲ ۱۷/۲/۱ پیکر مبنی زمان‌سنج ثابتهای
۵۴۲ ۱۷/۲/۲ ویژگی‌های زمان‌سنج ثابتهای
۵۴۲ ۱۷/۲/۳ عملکرد زمان‌سنج ثابتهای
۵۴۳ ۱۷/۲/۴ توضیح ثبات‌های زمان‌سنج ثابتهای
۵۴۴ ۱۷/۲/۵ مثال‌هایی در مورد محاسبه‌ی زمانی زمان‌سنج
۵۴۵ ۱۷/۳ زمان‌سنج نگهبان با بازه زمانی (WWDT)
۵۴۵ ۱۷/۳/۱ ویژگی‌های زمان‌سنج نگهبان

۵۴۵ کاربردهای زمان‌سنج نگهبان	۱۷/۳/۲
۵۴۵ توضیحاتی در مورد زمان‌سنج نگهبان	۱۷/۳/۳
۵۴۶ توضیحاتی در مورد ثبات‌های زمان‌سنج نگهبان	۱۷/۳/۴
۵۴۹ مثال‌هایی از زمان‌بندی زمان‌سنج نگهبان	۱۷/۳/۵
۵۵۱ تنظیم پهنای ضربان (PWM) و کنترل موتور	۱۸
۵۵۲ تنظیم پهنای ضربان (PWM)	۱۸/۱
۵۵۲ پیکربندی PWM	۱۸/۱/۱
۵۵۲ ویژگی‌های PWM	۱۸/۱/۲
۵۵۲ توضیحاتی در مورد PWM	۱۸/۱/۳
۵۵۴ شکل موج‌هایی با کنترل تک لبه و دو لبه	۱۸/۱/۴
۵۵۵ خروجی‌های PWM کنترل شده با تک لبه	۱۸/۱/۴/۱
۵۵۵ خروجی‌های PWM کنترل شده با دو لبه	۱۸/۱/۴/۲
۵۵۵ توضیح پایه‌های PWM	۱۸/۱/۵
۵۵۶ توضیح ثبات‌های PWM	۱۸/۱/۶
۵۶۲ کنترل موتور با PWM (MCPWM)	۱۸/۲
۵۶۲ پیکربندی کنترل موتور با PWM	۱۸/۲/۱
۵۶۲ عملکرد کلی کنترل موتور با PWM	۱۸/۲/۴
۵۶۲ توضیحاتی در مورد کنترل موتور با PWM	۱۸/۲/۲
۵۶۳ توضیحاتی در مورد پایه‌های کنترل موتور با PWM	۱۸/۲/۳
۵۶۴ توضیح ثبات‌های کنترل موتور با PWM	۱۸/۲/۵
۵۷۶ عملکرد PWM	۱۸/۲/۶
۵۷۶ تنظیم پهنای ضربان	۱۸/۲/۶/۱
۵۷۸ شبه ثبات‌ها و بمرورسانی	۱۸/۲/۶/۲
۵۷۸ توقف سریع (ABORT)	۱۸/۲/۶/۳
۵۷۸ رویدادهای اندازه‌گیری	۱۸/۲/۶/۴
۵۷۹ شمارش رویداد خارجی (حالت شمارنده)	۱۸/۲/۶/۵
۵۷۹ حالت سه فاز AC	۱۸/۲/۶/۶
۵۷۹ حالت سه فاز DC	۱۸/۲/۶/۷
۵۸۰ وقفه‌ها	۱۸/۲/۶/۸
۵۸۱ ضربان بلادرنگ (RTC)	۱۹
۵۸۲ پیکربندی ضربان بلادرنگ (RTC)	۱۹/۱
۵۸۲ ویژگی‌های RTC	۱۹/۲
۵۸۲ توضیحاتی در مورد کاربردهای RTC	۱۹/۳
۵۸۳ توضیح پایه‌های RTC	۱۹/۴
۵۸۳ توضیح ثبات‌های RTC	۱۹/۵
۵۸۴ وقفه‌های RTC	۱۹/۵/۱
۵۸۴ ثبات‌های متفرقه	۱۹/۵/۲
۵۸۶ ثبات‌های گروهی زمان	۱۹/۵/۳
۵۸۷ ثبات‌های شمارنده زمان	۱۹/۵/۴
۵۸۸ فرآیند تنظیم	۱۹/۵/۵
۵۸۹ ثبات‌های همه‌منظوره	۱۹/۵/۶
۵۹۰ ثبات‌های هشدار	۱۹/۵/۷
۵۹۱ ناظر و ثبت‌کننده رویدادها	۲۰
۵۹۲ پیکربندی بخش رویدادها	۲۰/۱
۵۹۲ ویژگی‌های بخش رویدادها	۲۰/۲
۵۹۲ توضیحاتی در مورد بخش رویدادها و کاربردهای آن	۲۰/۳
۵۹۳ توضیح پایه‌های ناظر/ثبت‌کننده رویداد	۲۰/۴
۵۹۳ توضیح ثبات‌های ناظر/ثبت‌کننده رویداد	۲۰/۵
۵۹۷ بخش آنالوگ	۲۱
۵۹۸ مبدل آنالوگ به دیجیتال (ADC)	۲۱/۱
۵۹۸ پیکربندی ADC	۲۱/۱/۱
۵۹۸ ویژگی‌های ADC	۲۱/۱/۲
۵۹۸ توضیحاتی در مورد پایه‌های ADC	۲۱/۱/۴
۵۹۸ توضیحاتی در مورد ADC	۲۱/۱/۳
۵۹۹ توضیح ثبات‌های ADC	۲۱/۱/۵

۶۰۲	۲۱/۱/۶ عملکرد ADC
۶۰۲	۲۱/۱/۶/۱ شروع تبدیل با تحریک سخت‌افزاری
۶۰۳	۲۱/۱/۶/۲ وقفه‌ها
۶۰۳	۲۱/۱/۶/۳ کنترل DMA
۶۰۳	۲۱/۲ مبدل دیجیتال به آنالوگ (DAC)
۶۰۳	۲۱/۲/۱ معماری DAC
۶۰۳	۲۱/۲/۲ پیکر بندی DAC
۶۰۴	۲۱/۲/۳ ویژگی‌های DAC
۶۰۴	۲۱/۲/۴ توضیحاتی در مورد پایه‌های DAC
۶۰۴	۲۱/۲/۵ توضیحاتی در مورد ثبات‌های DAC
۶۰۵	۲۱/۲/۶ عملکرد DAC
۶۰۵	۲۱/۲/۶/۱ شمارنده DMA
۶۰۵	۲۱/۲/۶/۲ نگهدارنده دوگانه
۶۰۶	۲۱/۳ مقایسه‌گر
۶۰۶	۲۱/۳/۱ پیکر بندی بخش مقایسه‌گر
۶۰۶	۲۱/۳/۲ ویژگی‌های بخش مقایسه‌گر
۶۰۶	۲۱/۳/۴ توضیحاتی در مورد پایه‌های بخش مقایسه‌گر
۶۰۷	۲۱/۳/۳ معماری بخش مقایسه‌گر
۶۰۷	۲۱/۳/۵ توضیح ثبات‌های بخش مقایسه‌گر
۶۱۳	۲۲ خطیابی و ردیابی
۶۱۴	۲۲/۱ تاریخچه خطیابی
۶۱۴	۲۲/۲ ویژگی‌های خطیابی در تراشه‌های LPC408x/7x
۶۱۴	۲۲/۳ توضیحاتی در مورد خطیابی و ردیابی
۶۱۵	۲۲/۴ توضیحاتی در مورد پایه‌های خطیابی و ردیابی
۶۱۶	۲۲/۵ خطیابی از طریق JTAG
۶۱۶	۲۲/۵/۱ تخریر مگاه
۶۱۷	۲۲/۵/۲ فرمان‌های مورد استفاده در خطیابی و ردیابی
۶۱۸	۲۲/۵/۳ کانکتورهای مورد استفاده برای خطیابی و ردیابی
۶۱۹	۲۲/۵/۴ فایل‌های BSDL
۶۲۰	۲۲/۶ کنترل‌کننده TAP
۶۲۱	۲۲/۷ تغییر آدرس حافظه در حالت خطیابی
۶۲۱	۲۲/۷/۱ ثبات کنترل تغییر آدرس حافظه (MEMMAP)
۶۲۲	۲۳ پیوست
۶۲۳	۲۳/۱ دسترسی خطی و غیرخطی
۶۲۳	۲۳/۲ بسته روی بسته (PoP)
۶۲۴	۲۳/۳ استاندارد شبه درین باز
۶۲۵	۲۳/۴ آشنایی با امکانات برد آموزشی کتاب (LPC4088-CB)
۶۲۶	۲۳/۴/۱ نقشه بخش تغذیه
۶۲۶	۲۳/۴/۲ نقشه بخش JTAG
۶۲۷	۲۳/۴/۳ نقشه بخش پردازنده
۶۲۸	۲۳/۴/۴ نقشه بخش حافظه‌های NAND و SDRAM
۶۲۸	۲۳/۴/۵ نقشه بخش حافظه‌های SPI و EEPROM
۶۲۸	۲۳/۴/۶ نقشه بخش حافظه uSD
۶۲۹	۲۳/۴/۷ نقشه بخش درگاه‌های USB
۶۲۹	۲۳/۴/۸ نقشه بخش مبدل USB به سریال
۶۲۹	۲۳/۴/۹ نقشه بخش بلوتوث و وای‌فای
۶۲۹	۲۳/۴/۱۰ نقشه بخش آنالوگ
۶۳۰	۲۳/۴/۱۱ نقشه بخش نمایشگر رنگی (LCD)
۶۳۰	۲۳/۴/۱۲ نقشه بخش پایه‌های خروجی
۶۳۱	۲۳/۵ نمودار کلی حافظه‌ها
۶۳۱	۲۳/۵/۱ نمودار کلی حافظه DDR با ظرفیت 128MB و گذرگاه داده x16
۶۳۲	۲۳/۵/۲ نمودار کلی حافظه DDR2 با ظرفیت 256MB و گذرگاه داده x16
۶۳۳	۲۳/۵/۳ نمودار کلی حافظه DDR3 با ظرفیت 512MB و گذرگاه داده x16
۶۳۴	۲۳/۵/۴ نمودار کلی حافظه DDR4 با ظرفیت 1GB و گذرگاه داده x16
۶۳۵	۲۳/۵/۵ نمودار کلی حافظه GDDR با ظرفیت 64MB و گذرگاه داده x32

۶۳۶	نمودار کلی حافظه GDDR2 با ظرفیت 32MB و گذرگاه داده x16	۲۳/۵/۶
۶۳۷	نمودار کلی حافظه GDDR3 با ظرفیت 64MB و گذرگاه داده x32	۲۳/۵/۷
۶۳۸	نمودار کلی حافظه GDDR4 با ظرفیت 64MB و گذرگاه داده x32	۲۳/۵/۸
۶۳۹	نمودار کلی حافظه GDDR5 با ظرفیت 1GB و گذرگاه داده x32	۲۳/۵/۹
۶۴۰	نمودار کلی حافظه GDDR5X با ظرفیت 1GB و گذرگاه داده x32	۲۳/۵/۱۰
۶۴۱	نمودار کلی حافظه LPDDR با ظرفیت 256MB و گذرگاه داده x32	۲۳/۵/۱۱
۶۴۲	نمودار کلی حافظه LPDDR2 با ظرفیت 512MB و گذرگاه داده x32	۲۳/۵/۱۲
۶۴۳	نمودار کلی حافظه LPDDR3 با ظرفیت 1GB و گذرگاه داده x32	۲۳/۵/۱۳
۶۴۴	نمودار کلی یکی از بخش‌های حافظه LPDDR4 با ظرفیت 384MB و گذرگاه داده x16	۲۳/۵/۱۴
۶۴۵	چینش پایه‌ها	۲۳/۵
۶۴۵	چینش پایه‌های تراشه DDR در بسته‌بندی TSOP-66 با پهنای گذرگاه داده x16	۲۳/۵/۱
۶۴۶	چینش پایه‌های تراشه DDR2 در بسته‌بندی FBGA-84 با پهنای گذرگاه داده x16	۲۳/۵/۲
۶۴۷	چینش پایه‌های تراشه DDR3 در بسته‌بندی FBGA-96 با پهنای گذرگاه داده x16	۲۳/۵/۳
۶۴۸	چینش پایه‌های تراشه DDR4 در بسته‌بندی FBGA-96 با پهنای گذرگاه داده x16	۲۳/۵/۴
۶۴۹	چینش پایه‌های تراشه GDDR1 در بسته‌بندی FBGA-144 با پهنای گذرگاه داده x32	۲۳/۵/۵
۶۵۰	چینش پایه‌های تراشه GDDR2 در بسته‌بندی TFBGA-84 با پهنای گذرگاه داده x16	۲۳/۵/۶
۶۵۱	چینش پایه‌های تراشه‌های GDDR3 و GDDR4 در بسته‌بندی FBGA-136 با پهنای گذرگاه داده x32	۲۳/۵/۷
۶۵۲	چینش پایه‌های تراشه GDDR5 در بسته‌بندی FBGA-170 با پهنای گذرگاه داده x32	۲۳/۵/۸
۶۵۳	چینش پایه‌های تراشه GDDR5X در بسته‌بندی FBGA-190 با پهنای گذرگاه داده x32	۲۳/۵/۹
۶۵۴	چینش پایه‌های تراشه LPDDR1 در بسته‌بندی FBGA-168 با پهنای گذرگاه داده x32	۲۳/۵/۱۰
۶۵۶	چینش پایه‌های تراشه LPDDR2 در بسته‌بندی FBGA-168 با پهنای گذرگاه داده x32	۲۳/۵/۱۱
۶۵۸	چینش پایه‌های تراشه LPDDR3 در بسته‌بندی FBGA-168 با پهنای گذرگاه داده x32	۲۳/۵/۱۲
۶۶۰	چینش پایه‌های تراشه LPDDR4 در بسته‌بندی FBGA-272 با پهنای گذرگاه داده x64	۲۳/۵/۱۳
۶۶۲	نوع بسته‌بندی	۲۳/۶
۶۶۲	شکل بسته‌بندی حافظه DDR به صورت TSOP-66	۲۳/۶/۱
۶۶۲	شکل بسته‌بندی حافظه DDR2 به صورت FBGA-84	۲۳/۶/۲
۶۶۲	شکل بسته‌بندی حافظه DDR3 به صورت FBGA-96	۲۳/۶/۳
۶۶۳	شکل بسته‌بندی حافظه DDR4 به صورت FBGA-96	۲۳/۶/۴
۶۶۳	شکل بسته‌بندی حافظه GDDR1 به صورت FBGA-144	۲۳/۶/۵
۶۶۳	شکل بسته‌بندی حافظه GDDR2 به صورت TFBGA-84	۲۳/۶/۶
۶۶۴	شکل بسته‌بندی حافظه‌های GDDR3 و GDDR4 به صورت FBGA-136	۲۳/۶/۷
۶۶۴	شکل بسته‌بندی حافظه GDDR5 به صورت FBGA-170	۲۳/۶/۸
۶۶۴	شکل بسته‌بندی حافظه GDDR5X به صورت FBGA-190	۲۳/۶/۹
۶۶۵	شکل بسته‌بندی حافظه‌های LPDDR1 و LPDDR2 به صورت FBGA-168	۲۳/۶/۱۰
۶۶۵	شکل بسته‌بندی حافظه LPDDR4 به صورت FBGA-272	۲۳/۶/۱۱

فصل

۵

آشنایی با نرم افزار کیل

نرم افزار کیل یک محیط برنامه نویسی و ویرایشگر برای میکروکنترلرها بوده و در سال ۱۹۸۲ توسط شرکت کیل تولید شده است. این نرم افزار در سال ۲۰۰۵ توسط شرکت ARM خریداری شد. شرکت ARM پس از خریداری این نرم افزار، تمرکز خود را در این نرم افزار بر روی میکروکنترلرهایی قرار داده که از هسته های این شرکت استفاده می کنند. این نرم افزار تمامی امکانات مورد نیاز برای برنامه نویسی میکروکنترلرهای ARM از جمله مترجم زبان های C و ++C، خطایاب، شبیه ساز و محیط IDE را فراهم می کند. این نرم افزار از محیط برنامه نویسی و ویرایشگر uVision5 و مترجم armcc (ARM C/C++ Compiler) استفاده می کند. نرم افزار کیل از بیش از ۵۶۰۰ میکروکنترلر مبتنی بر هسته ARM که توسط بیش از ۳۲ شرکت مختلف تولید شده اند، پشتیبانی می کند.

برخی از قابلیت های نرم افزار کیل:

- پشتیبانی کامل از هسته های ARM7، ARM9، Cortex-M و غیره
- پشتیبانی از زبان های برنامه نویسی C، ++C و اسمبلی
- دارای محیط توسعه یکپارچه شامل محیط برنامه نویسی و ویرایشگر uVision5، شبیه ساز و خطایاب پیشرفته
- دارای مترجم پیشرو در صنعت با نام armcc
- دارای سیستم عامل بلادرنگ اختصاصی (RTX) به همراه کد منبع آن
- کلاس های برنامه نویسی مختلف برای کار با شبکه TCP/IP دستگاه های USB و غیره
- مجهز به کتابخانه های پیشرفته GUI برای کمک در نوشتن برنامه های گرافیکی
- پشتیبانی از خطایاب ها و برنامه ریزهای مختلفی همچون ST-Link، J-Link و غیره جهت برنامه ریزی میکروکنترلرها و خطایابی برنامه ها
- دارای مثال های آماده و گوناگون جهت آشنایی و یادگیری سریع تر

نرم افزار کیل دارای چهار نوع ویرایش است:

- MDK-Lite: نرم افزار کیل پس از نصب، به صورت پیش فرض دارای این ویرایش است. این ویرایش دارای محدودیت کد به اندازه 32KB بوده و برای بردها و تجهیزات آموزشی و پروژه های کوچک استفاده می شود.
- MDK-Essential: در این ویرایش فقط می توان برای هسته های سری Cortex-M برنامہ نوشت و از برنامه نویسی امن در هسته های Cortex-M23 و Cortex-M33 حمایت نمی شود.
- MDK-Plus: این ویرایش علاوه بر ویژگی های ویرایش "MDK-Essential"، دارای کتابخانه هایی برای مواردی چون آدرس دهی IPv4، برنامه های گرافیکی، حالت میهمان USB و مدیریت فایل ها و پوشه ها است. در این ویرایش می توان برای هسته های سری ARM7، ARM9، هسته های امن، Cortex-M و Cortex-R4 برنامه نوشت و از برنامه نویسی امن در هسته های Cortex-M23 و Cortex-M33 نیز حمایت می شود.
- MDK-Professional: این ویرایش علاوه بر ویژگی های ویرایش "MDK-Plus"، از آدرس دهی IPv6، شبکه اینترنت اشیا و حالت میزبان USB و نیز از برنامه نویسی هسته هایی با معماری ARMv8-M حمایت می کند. نرم افزار کیل در این ویرایش دارای تمامی امکانات ارائه شده توسط شرکت ARM است.

۵/۱ چگونگی دریافت و نصب نرم افزار کیل و بسته های آن

۵/۱/۱ چگونگی دریافت نرم افزار کیل

نگارش ۵/۲۶ این نرم افزار در DVD کتاب و در آدرس "Compiler/Keil" و با نام "MDK526.EXE" قرارداد شده است. در صورتی که می خواهید آخرین نگارش آن را دریافت کنید، ابتدا به آدرس "https://www.keil.com/demo/eval/arm.htm" بروید. در این آدرس با صفحه ای مانند شکل بعد روبرو می شوید، که می بایست موارد مشخص شده را پر کنید.

armKEIL
Search Keil...

Home / Product Downloads

MDK-ARM

MDK-ARM Version 5.26
Version 5.26

Complete the following form to download the Keil software development tools.

Enter Your Contact Information Below

First Name:

Last Name:

E-mail:

Company:

Address:

City:

State/Province:

Zip/Postal Code:

Country:

Phone:

Send me e-mail when there is a new update.

NOTICE:
If you select this check box, you will receive an e-mail message from Keil whenever a new update is available. If you don't wish to receive an e-mail notification, don't check this box.

Which device are you using?
(eg, STM32)

Arm will process your information in accordance with the Evaluation section of our Privacy Policy.

By ticking this box you confirm you have read and accept our Privacy Policy and indicate your consent to receiving marketing communications from Arm. Please visit our Subscription Centre to manage your marketing preferences or unsubscribe from future communications.

شکل ۱. صفحه ورود مشخصات برای دریافت فایل نرم‌افزار کایل

با زدن بر روی کلید "Submit" به صفحه بعد می‌رسیم. در این صفحه با زدن بر روی نام فایل و دادن آدرس دلخواه برای ذخیره شدن بر روی رایانه تان، دریافت نرم‌افزار کایل آغاز می‌شود.

Home / Product Downloads

MDK-ARM

MDK-ARM Version 5.26
Version 5.26

- Review the hardware requirements before installing this software.
- Note the limitations of the evaluation tools.
- Further installation instructions for MDK5

(MD5:3938932e4e7ba7ee3b579f9a6a270710)

To install the MDK-ARM Software...

- Right-click on **MDK526.EXE** and save it to your computer.
- PDF files may be opened with Acrobat Reader.
- ZIP files may be opened with PKZIP or WINZIP.

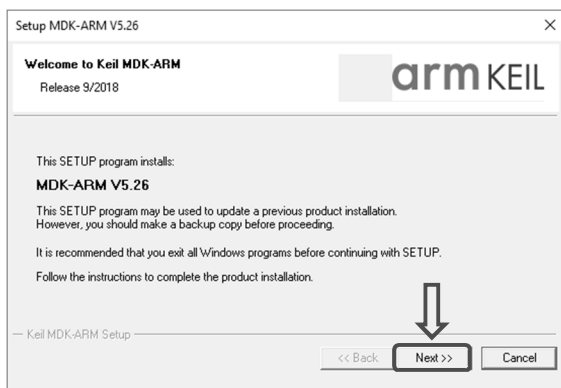
MDK526.EXE (915,212K)
Monday, September 10, 2018

- If you are evaluating the tools, be sure to request a quote for the full version of the tools.

شکل ۲. صفحه دریافت فایل نرم افزار کیل

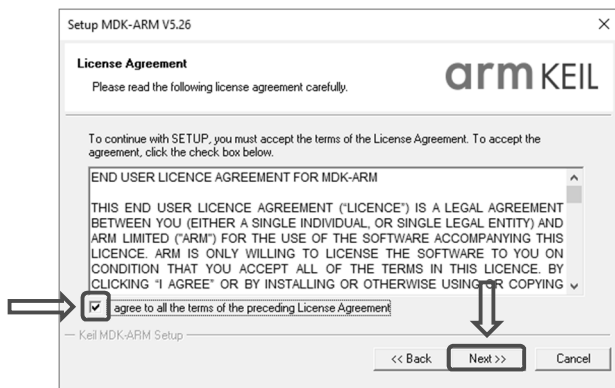
۵/۱/۲ چگونگی نصب نرم افزار کیل

با اجرای فایل نگارش ۵/۲۶ (آدرس آن در بخش قبل ذکر شده) یا اجرای فایل آخرین نگارش دریافت شده، به پنجره روبرو می‌رسیم.

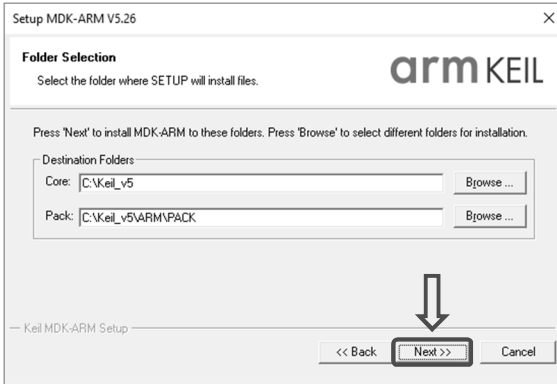


شکل ۳. پنجره نصب نرم افزار کیل

بر روی کلید Next می‌زنیم تا پنجره قوانین و شرایط استفاده، ظاهر شود.



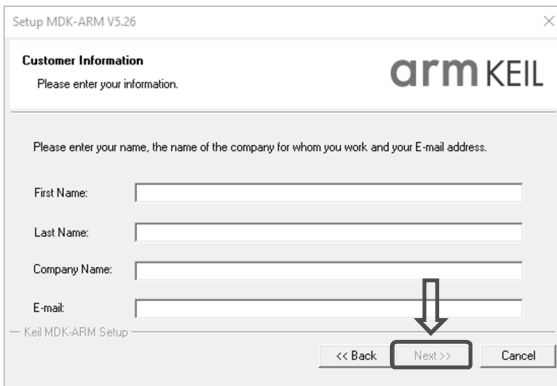
شکل ۴. پنجره قوانین و شرایط استفاده



با انتخاب جعبه بررسی، قوانین و شرایط استفاده را پذیرفته و سپس بر روی کلید Next می‌زنیم تا پنجره محل نصب نرم‌افزار کایل ظاهر شود.

شکل ۵. پنجره محل نصب نرم‌افزار کایل

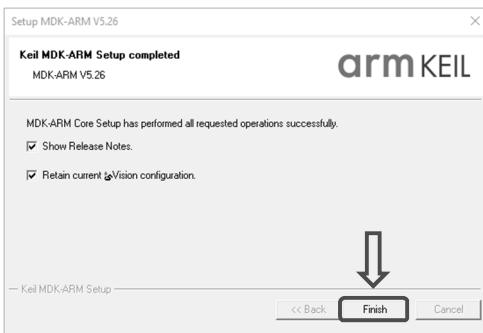
تذکر: شما می‌توانید آدرس محل نصب نرم‌افزار کایل را تغییر دهید اما توصیه می‌کنیم آنرا تغییر ندهید زیرا بسیاری از برنامه‌های نوشته شده توسط دیگران از همین آدرس پیش‌فرض برای دسترسی به بعضی از فایل‌های مورد نیازشان استفاده می‌کنند و در صورت تغییر این آدرس، دیگر نمی‌توانند به آن فایل‌ها دسترسی داشته باشند و در هنگام ترجمه آن برنامه با خطا مواجه خواهید شد.



پس از تعیین آدرس، بر روی کلید Next می‌زنیم تا پنجره اطلاعات شخصی همانند شکل روبرو ظاهر شود.

شکل ۶. پنجره اطلاعات شخصی

هر چهار مورد را به صورت دلخواه پر کرده و سپس بر روی کلید Next می‌زنیم تا فرآیند نصب آغاز شود.

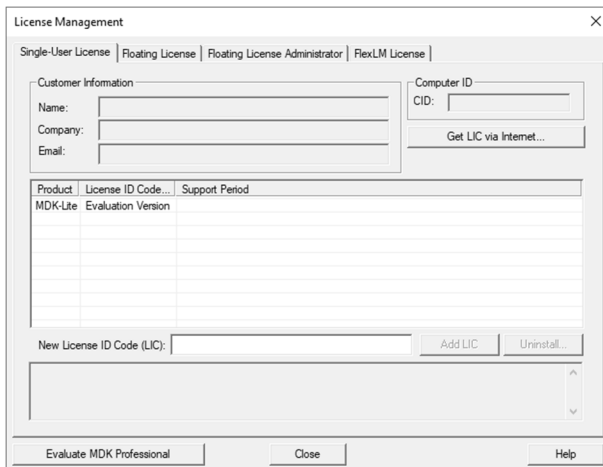


شکل ۷. پنجره شروع فرآیند نصب (سمت راست) و پنجره پایان فرآیند نصب (سمت چپ)

با پایان فرآیند نصب بر روی کلید Finish می‌زنیم.

۵/۱/۳ چگونگی کرک نرم‌افزار کایل

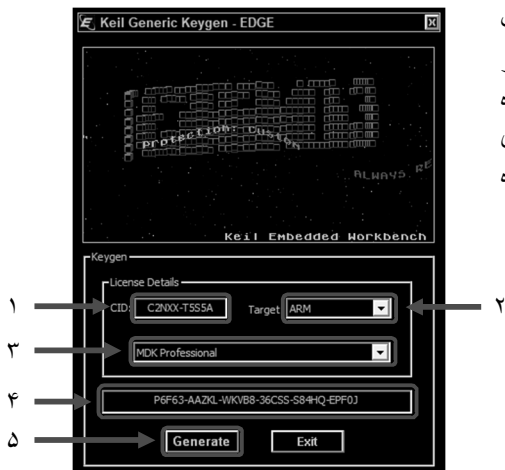
نرم‌افزار کایل به صورت پیش فرض دارای محدودیت ترجمه برنامه تا حجم 32KB می‌باشد که برای برداشتن این محدودیت می‌بایست به آن گواهینامه دهیم. برای این کار ابتدا نرم‌افزار کایل را طبق بخش ۵/۱/۴ اجرا می‌کنیم. سپس



از منوی File گزینه "License Managment" را انتخاب می‌کنیم تا پنجره روبرو باز شود.

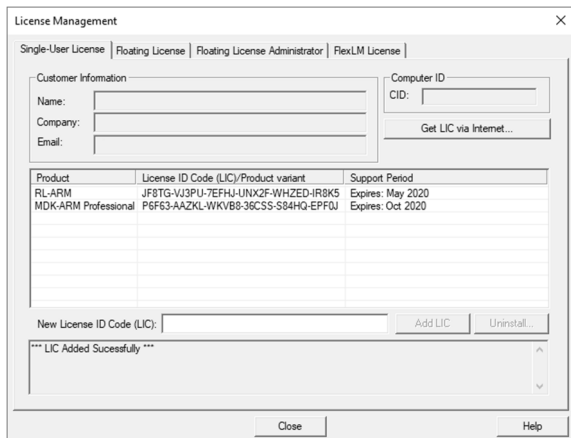
شکل ۸. پنجره مدیریت گواهینامه‌ها

در برگه "Single-User License" از قسمت "Computer ID" بخش CID را کپی می‌کنیم. سپس در DVD کتاب و در آدرس "Compiler/Keil" فایل فشرده "KeyGen v5.xx.7z" را اجرا می‌کنیم. در این فایل، دو فایل ایجادکننده گواهینامه وجود دارد که یکی از آنها را اجرا کرده و در نتیجه پنجره‌ای همانند شکل روبرو باز می‌شود.



شکل ۹. پنجره فایل ایجادکننده گواهینامه

CID را در بخش (۱) کپی کرده و در بخش (۲) نوع میکرو را ARM انتخاب می‌کنیم. در بخش (۳) نوع گواهینامه را "MDK Professional" انتخاب کرده و بر روی کلید Generate می‌زنیم (بخش ۵). در بخش (۴) گواهینامه موردنظر ما ایجاد می‌شود که آن را کپی می‌کنیم و در پنجره شکل قبل و در قسمت "New License ID Code (LIC)" کپی می‌کنیم و سپس کلید "Add LIC" را می‌زنیم.



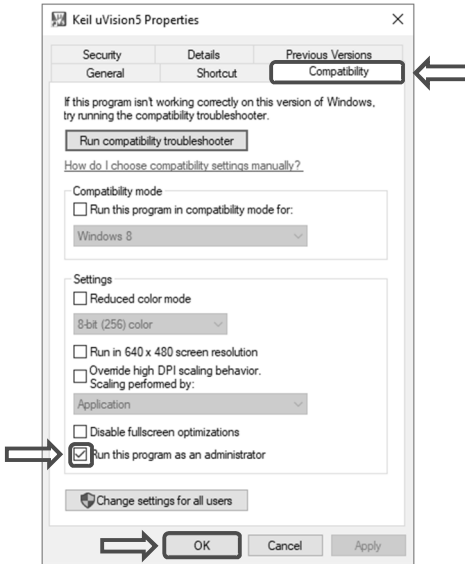
در صورت نیاز به سیستم‌عامل بلادرنگ اختصاصی (RTX) شرکت ARM، نرم‌افزار کیل به گواهینامه آن نیاز داشته و در نتیجه برای داشتن گواهینامه آن، باید همین کار را تکرار کنیم. برای این کار، در بخش (۳) عبارت "RL/ARTX" را انتخاب و بقیه مراحل را مثل قبل تکرار می‌کنیم. بدین ترتیب، پنجره مدیریت گواهینامه‌ها باید به صورت روبرو درآمده باشد.

شکل ۱۰. پنجره مدیریت گواهینامه‌ها پس از دادن گواهینامه به نرم‌افزار کیل

پس از این مرحله نرم‌افزار کیل به صورت کامل کرک شده و بدون محدودیت قابل استفاده خواهد بود.

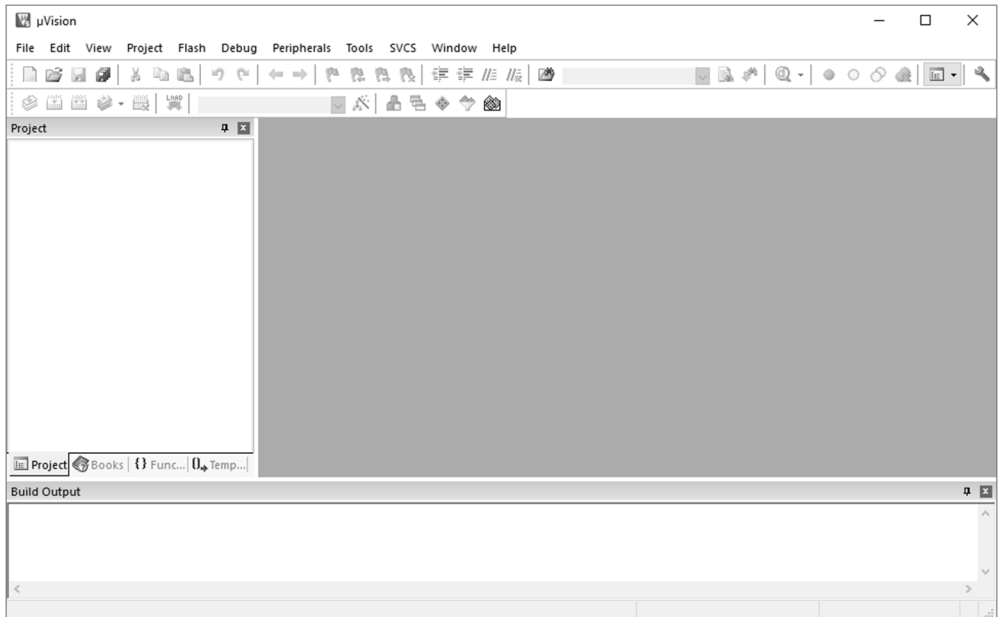
۵/۱/۴ چگونگی اجرای نرم‌افزار کیل

اگر برای اولین بار می‌خواهید نرم‌افزار کیل را اجرا کنید ابتدا بر روی میانبر نرم‌افزار (ایجاد شده پس از نصب بر روی دسکتاپ با نام "Keil uVision5") راست کلیک کرده و از پنجره باز شده و در انتهای آن، گزینه Properties را انتخاب کنید. در این هنگام پنجره‌ای همانند شکل روبرو باز می‌شود.



شکل ۱۱. پنجره Properties

در پنجره باز شده به سربرگ Compatibility رفته و در انتهای آن جعبه بررسی "Run this program as an administrator" را انتخاب و در پایان بر روی کلید OK می‌زنیم. سپس بر روی میانبر نرم‌افزار می‌زنیم تا نرم‌افزار اجرا شود. نرم‌افزار کیل همانند شکل زیر باز می‌شود.

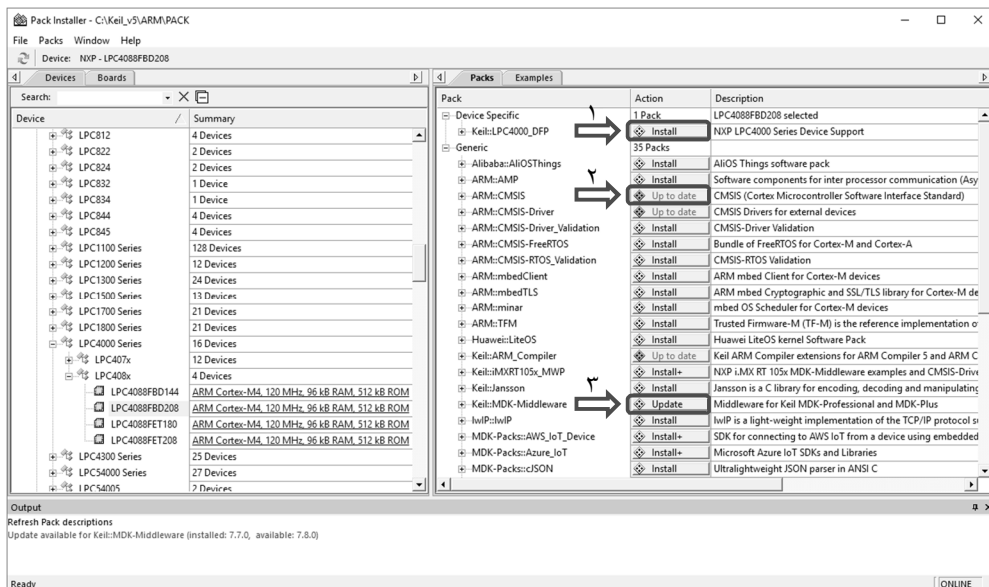


شکل ۱۲. نرم‌افزار کیل

۵/۱/۵ چگونگی دریافت، نصب و بروزرسانی بسته‌ها در نرم‌افزار کیل

برای کاهش حجم نرم‌افزار کیل، شرکت ARM برای هر سری از تراشه‌های هر شرکتی، کتابخانه‌ها و فایل‌های مورد نیاز آن سری را در یک بسته نرم‌افزاری جداگانه‌ای (با پسوند ".pack") قرار داده است. کار با این بسته‌های نرم‌افزاری از طریق پنجره "Pack Installer" انجام می‌شود.

این پنجره پس از نصب نرم‌افزار کیل به صورت خودکار باز می‌شود. در دیگر زمان‌ها برای باز کردن این پنجره باید از منوی Project، گزینه Manage و سپس گزینه "Pack Installer" را انتخاب کنید. همچنین این پنجره را می‌توان با انتخاب ابزار "Pack Installer" (اولین ابزار از سمت راست) در نوار ابزار Build، باز کرد. در این حالت‌ها پنجره‌ای همانند شکل زیر باز می‌شود.



شکل ۱۳. پنجره "Pack Installer"

این پنجره از دو نیمه تشکیل شده که از طریق نیمه سمت چپ، تراشه موردنظر انتخاب شده و از طریق نیمه سمت راست می‌توان بسته‌های دلخواه را دریافت و نصب یا بروزرسانی کرد. توضیحات سه قسمت مشخص شده در شکل بالا به صورت زیر است:

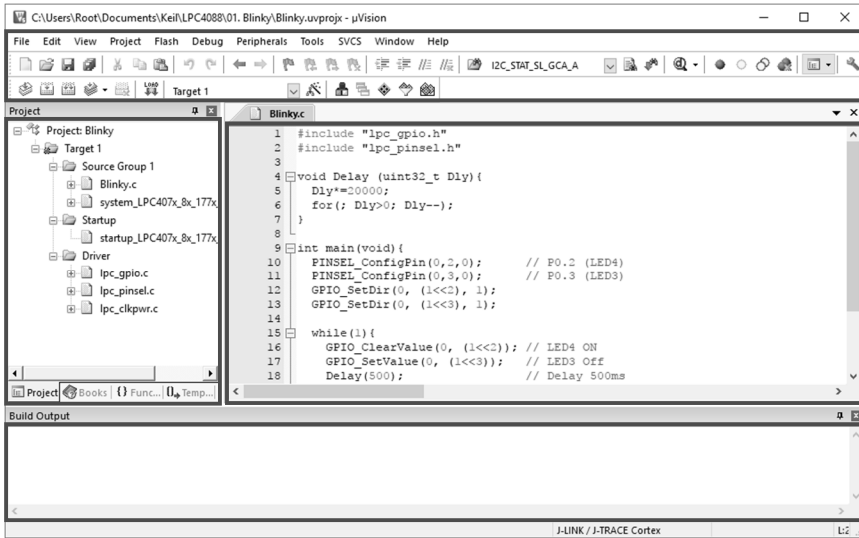
- **گزینه (۱) دریافت و نصب:** بسته موردنظر دریافت و نصب می‌شود.
- **گزینه (۲) بروزرودن:** نشان دهنده آن است که برای آن مورد، جدیدترین بسته نرم‌افزاری نصب شده است.
- **گزینه (۳) بروزرسانی:** نشان دهنده آن است که بسته نرم‌افزاری نصب شده برای آن مورد، قدیمی بوده اما نگارش بروزتری برای آن وجود دارد و می‌تواند توسط این گزینه، بروزرسانی شود.

۵/۲ معرفی بخش‌های مختلف نرم‌افزار کیل

۵/۲/۱ معرفی بخش‌های کلی نرم‌افزار کیل

نرم‌افزار کیل از چهار بخش کلی زیر تشکیل شده و نمای کلی آن در شکل بعد آمده است.

- **منو و نوار ابزارها:** دارای گزینه‌ها و ابزارهایی برای مدیریت نرم‌افزار کیل
- **ویرایشگر:** برای نوشتن برنامه و ویرایش آن
- **نمودار درختی:** دارای نمودار درختی پروژه شامل پوشه‌ها و فایل‌های مختلف
- **پیام‌نما:** برای ارتباط با برنامه‌نویس و نمایش پیام‌ها



شکل ۱۴. نمای کلی نرم افزار کایل

۵/۲/۲ توضیحاتی در مورد منوی نرم افزار کایل

منوی نرم افزار کایل شامل موارد زیر است:

- **File:** شامل گزینه‌هایی برای ایجاد، ذخیره و باز و بسته‌کردن فایل‌ها و غیره
- **Edit:** شامل گزینه‌هایی برای ویرایش، نشانه‌گذاری، جستجو و غیره در برنامه و همچنین تنظیم کایل
- **View:** شامل گزینه‌هایی برای نمایش/عدم نمایش هر کدام از پنجره‌ها، نوار ابزارها و نوار وضعیت در کایل
- **Project:** شامل گزینه‌هایی برای ایجاد، باز و بسته‌کردن، ترجمه و مدیریت پروژه
- **Flash:** شامل گزینه‌هایی برای انتقال فایل hex برنامه به تراشه، پاک‌کردن تراشه و مدیریت ارتباط با تراشه
- **Debug:** شامل گزینه‌هایی برای شبیه‌سازی/خطایابی تراشه
- **Peripheral:** شامل گزینه‌هایی برای نشان دادن اطلاعات لحظه‌ای مربوط به بخش‌های داخلی تراشه در هنگام شبیه‌سازی/خطایابی
تذکر: این بخش در حالت عادی غیرفعال است و با فعال‌شدن شبیه‌سازی/خطایابی، این بخش نیز فعال می‌شود.
- **Tools:** شامل گزینه‌هایی برای پیکربندی و اجرای نرم افزارهای جانبی (مربوط به دیگر شرکت‌ها) در کایل
- **SVCS:** پیکربندی و ویرایشگر کایل برای استفاده از نرم افزار کنترل نگارش
- **Window:** شامل گزینه‌هایی برای مدیریت بخش ویرایشگر
- **Help:** شامل گزینه‌هایی مانند راهنمای نرم افزار، بررسی بروزرسانی و غیره

۵/۲/۳ توضیحاتی در مورد نوار ابزارهای نرم افزار کایل

نرم افزار کایل دارای دو نوار ابزار **File** و **Build** است.

نوار ابزار **File** به صورت زیر بوده و دارای ابزارهایی است که کارکرد آنها در ادامه آمده است:



- **New:** از این ابزار برای ایجاد یک فایل نوشتاری جدید استفاده می‌شود. کلید میانبر آن "Ctrl+N" است.
- **Open:** از این ابزار برای باز کردن فایل نوشتاری (شامل فایل‌های منبع، سرآیند، متن و غیره)، پروژه، ایستگاه کاری و غیره استفاده می‌شود. کلید میانبر آن "Ctrl+O" است.
- **Save** و **Save all:** از این دو ابزار برای ذخیره‌سازی یک یا همه فایل‌های نوشتاری استفاده می‌شود. کلید میانبر آن "Ctrl+S" است.
- **Cut، Copy** و **Paste:** از این ابزارها برای برداشتن، کپی کردن و چسباندن متن در فایل نوشتاری، استفاده می‌شود. کلیدهای میانبر آنها به ترتیب "Ctrl+X"، "Ctrl+C" و "Ctrl+V" است.

- **Undo و Redo:** در صورت ایجاد تغییر در برنامه، ابزار Undo، آخرین تغییر انجام شده در برنامه را حذف و ابزار Redo، آخرین تغییر را به برنامه بازمی‌گرداند. کلیدهای میانبر آنها به ترتیب "Ctrl+Z" و "Ctrl+Y" است.
- **Navigate Tools:** از این ابزارها برای رفتن و برگشتن به موقعیت‌های بعدی و قبلی استفاده می‌شود. کلیدهای میانبر آنها به ترتیب "Ctrl+Shift+-" و "Ctrl+-" است.
- **Bookmark Tools:** از این چهار ابزار برای مدیریت نشانه‌گذاری خط‌های برنامه استفاده می‌شود. اولین ابزار از سمت چپ برای نشانه‌گذاری/نشانه‌برداری یک خط از برنامه استفاده می‌شود. از دو ابزار وسط برای جابجایی بین خط‌های دارای نشانه استفاده می‌شود. اولین ابزار از سمت راست برای حذف همه نشانه‌ها از برنامه استفاده می‌شود. کلیدهای میانبر آنها به ترتیب "Ctrl+F2"، "Shift+F2"، "F2" و "Ctrl+Shift+F2" است.
- **تذکر:** با نشانه‌گذاری یک خط، یک مربع آبی رنگ به سمت چپ خط اضافه می‌شود.
- **Indent Selection و Unindent Selection:** با استفاده از این دو ابزار می‌توان متن انتخاب شده را به اندازه یک Tab به راست یا چپ حرکت داد.
- **Comment Selection و Uncomment Selection:** با استفاده از این دو ابزار می‌توان متن انتخاب شده را به توضیحات تبدیل کرده یا از حالت توضیحات خارج کرد.
- **Find a File:** با استفاده از این ابزارها می‌توان در یک فایل و یا همه فایل‌ها (کل پروژه) به جستجوی متن، دستور و غیره پرداخت. کلیدهای میانبر آنها "Ctrl+Shift+F"، "Ctrl+F" و "Ctrl+I" است.
- **Debug Session:** با استفاده از این ابزار می‌توان به حالت شبیه‌سازی برنامه وارد یا از آن خارج شد. کلید میانبر آن "Ctrl+F5" است.
- **تذکر:** در صورت ناقص بودن و یا درست ترجمه نشدن برنامه (به دلیل وجود خطا) این ابزار غیرفعال خواهد بود.
- **Breakpoint Tools:** از این چهار ابزار برای مدیریت نقطه‌توقف‌ها در برنامه استفاده می‌شود. اولین ابزار از سمت چپ برای گذاشتن/برداشتن نقطه‌توقف (F9)، دومین ابزار برای فعال/غیرفعال کردن نقطه‌توقف موردنظر (Ctrl+F9)، سومین ابزار برای غیرفعال کردن همه نقطه‌توقف‌ها و چهارمین ابزار برای حذف همه نقطه‌توقف‌ها از برنامه (Ctrl+Shift+F9) استفاده می‌شود.
- **Project Window:** این ابزار دارای بیشتر گزینه‌های موجود در منوی View است.
- **Configuration:** این ابزار برای تنظیم ویرایشگر، رنگ و قلم، کلیدهای میانبر، الگوها و غیره استفاده می‌شود.

نوار ابزار Build به صورت زیر بوده و دارای ابزارهایی است که کارکرد آنها در ادامه آمده است:



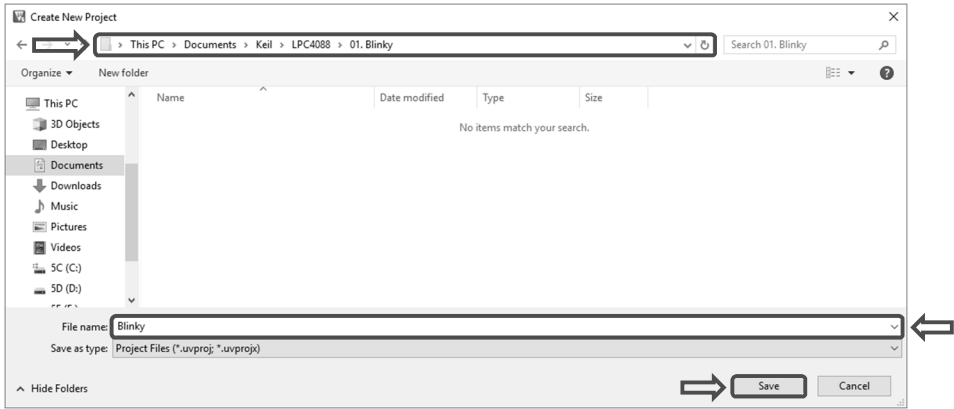
- **Translate:** از این ابزار برای ترجمه فایل در حال نمایش در بخش ویرایشگر، استفاده می‌شود. کلید میانبر آن "Ctrl+F7" است. **تذکر:** این ابزار فقط یک فایل را ترجمه می‌کند.
- **Build:** این ابزار فقط فایل‌های تغییر یافته را ترجمه کرده و در نهایت فایل hex برنامه را ایجاد می‌کند. کلید میانبر آن "F7" است.
- **Rebuild:** این ابزار همه فایل‌های برنامه را ترجمه کرده و در نهایت فایل hex برنامه را ایجاد می‌کند.
- **Batch Build:** از این ابزار برای ترجمه برنامه‌ها به صورت گروهی استفاده می‌شود. این ابزار دارای چهار گزینه است. برای استفاده از این ابزار ابتدا باید چهارمین گزینه (Batch Setup) را انتخاب کرده و از نمودار درختی موجود در پنجره باز شده، برنامه‌های مورد نظر را انتخاب و کلید Close را بزنید. در این حالت اولین گزینه (Batch Build) در تمامی برنامه‌های انتخاب شده فقط فایل‌های تغییر یافته را ترجمه کرده و در نهایت فایل hex همه برنامه‌ها را ایجاد می‌کند. دومین گزینه (Batch Rebuild) تمامی برنامه‌های انتخاب شده را به صورت کامل ترجمه کرده و در نهایت فایل hex همه برنامه‌ها را ایجاد می‌کند. سومین گزینه (Batch Clean) فایل‌های اضافی تولید شده در هنگام ترجمه برنامه توسط کیل در پوشه "Objects" و "Listings" را پاک می‌کند.
- **Stop Build:** از این ابزار برای متوقف کردن ترجمه در حال انجام، استفاده می‌شود.
- **Download:** از این ابزار برای انتقال فایل hex به برد مدار چاپی استفاده می‌شود. کلید میانبر آن "F8" است.
- **Select Target:** از این ابزار برای انتخاب پروژه موردنظر در محیط ویرایشگر، اگر چند پروژه با هم باز باشند، استفاده می‌شود.
- **Options for Target:** از این ابزار برای باز کردن پنجره "Options for Target" استفاده می‌شود.
- **File Extensions:** از این ابزار برای باز کردن پنجره "Manage Project Items" و مدیریت فایل‌ها در یک پروژه استفاده می‌شود.
- **Manage Multi-Project Workspace:** از این ابزار برای مدیریت برنامه‌ها در یک ایستگاه کاری استفاده می‌شود.

- Manage Run-Time Environment: از این ابزار برای مدیریت امکانات مورد نیاز برنامه استفاده می‌شود.
- Select Software Packs: از این ابزار برای مدیریت بسته‌های استفاده شده در برنامه استفاده می‌شود.
- Pack Installer: از این ابزار برای باز کردن پنجره "Pack Installer" و مدیریت بسته‌های نصب شده در نرم افزار کایل استفاده می‌شود.

۵/۳ ایجاد پروژه و تنظیمات آن در نرم افزار کایل

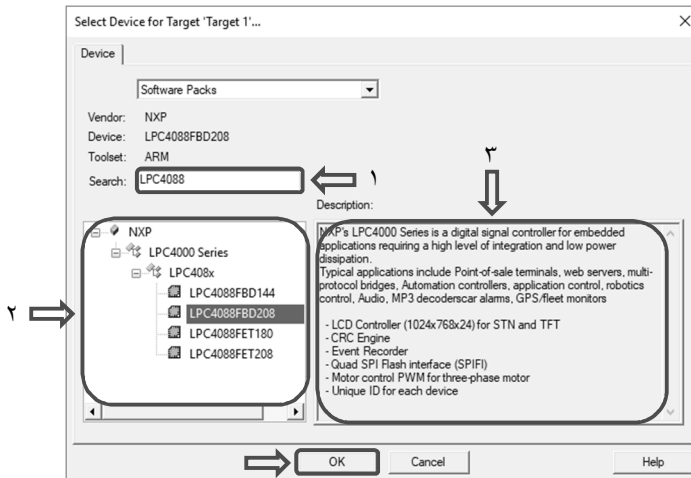
۵/۳/۱ چگونگی ایجاد پروژه

برای ایجاد یک پروژه باید از منوی Project، گزینه "New uVision Project" را انتخاب کنیم. در این زمان پنجره "Create New Project" مطابق شکل زیر باز می‌شود.



شکل ۱۵. پنجره "Create New Project"

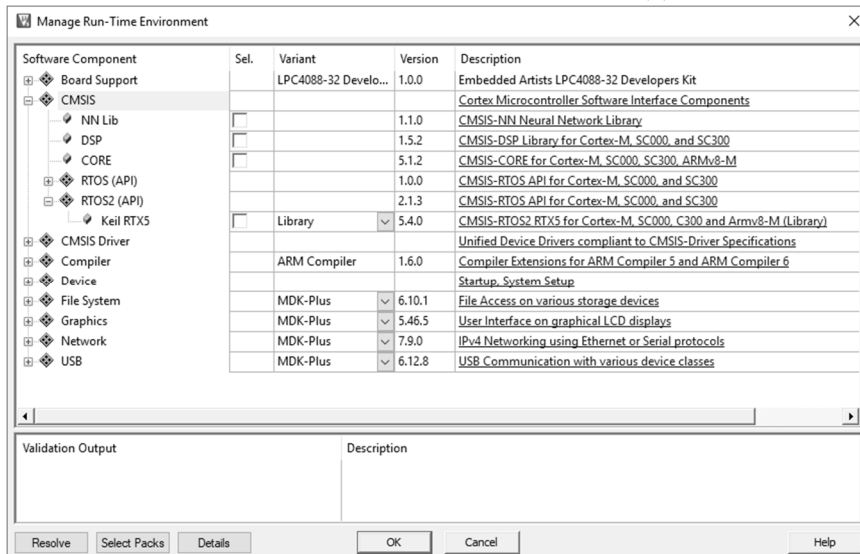
در این پنجره مسیر پروژه را انتخاب کرده، برای پروژه نامی در نظر گرفته و سپس آنرا ذخیره می‌کنیم. پس از ذخیره شدن پروژه، پنجره شکل بعد به صورت خودکار باز می‌شود.



شکل ۱۶. پنجره انتخاب تراشه

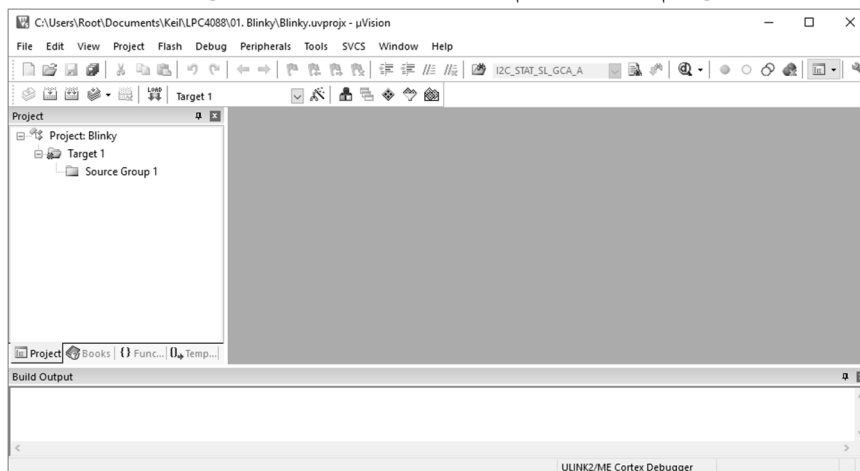
پنجره شکل بالا، پنجره انتخاب تراشه بوده و بنابراین برای انتخاب تراشه یکی از دو کار زیر را می‌توان انجام داد ۱. جستجوی نام کامل تراشه و سپس انتخاب آن در قاب (۲)، ۲. نوشتن بخشی از نام تراشه در قاب (۱) و انتخاب نام کامل تراشه در قاب (۲). روش دوم ساده‌تر بوده و در هر دو حالت پس از انتخاب نام کامل تراشه، توضیحات مختصری در

مورد تراشه انتخاب شده در قاب (۳) خواهد آمد. در انتهای کار کلید OK را زده و در نتیجه پنجره زیر باز می‌شود.



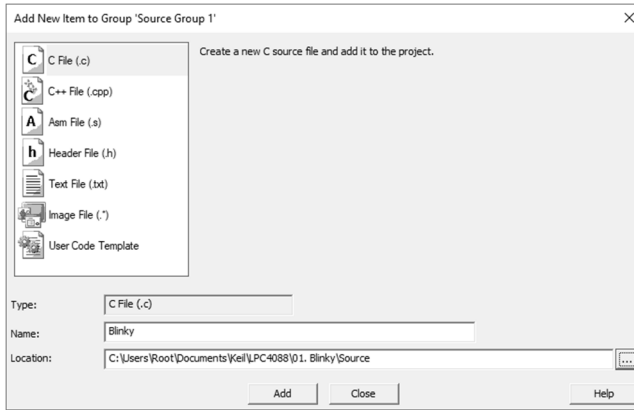
شکل ۱۷. پنجره انتخاب امکانات مورد نیاز

در سمت چپ شکل بالا، نمودار درختی امکانات قرار داشته که می‌توان با استفاده از آن، امکانات مورد نظر خود را مشخص و از طریق ستون Sel، آنها را انتخاب کرد. هر امکاناتی که در این پنجره انتخاب شود، به پروژه اضافه خواهد شد. به دلیل اینکه پروژه ما یک پروژه چشمک زن ساده است بنابراین به هیچ‌کدام از این امکانات نیازی نداریم و فقط کلید OK را می‌زنیم و در نتیجه نرم‌افزار کیل به صورت زیر باز می‌شود.



شکل ۱۸. پروژه ایجادشده در نرم‌افزار کیل

در این پروژه هیچ فایل‌ی وجود ندارد و نمودار درختی آن خالی است و برای نوشتن برنامه باید فایل‌های مربوط به پروژه در آن ایجاد یا به آن اضافه شوند. در صورت موجود بودن فایل اصلی پروژه (فایل دارای تابع main)، آن را به پروژه اضافه کرده و در صورت موجود نبودن، آن را ایجاد می‌کنیم. فرض می‌کنیم فایل اصلی پروژه وجود ندارد بنابراین در نمودار درختی بر روی پوشه "Source Group 1" راست کلیک کرده و گزینه "Add new" را انتخاب می‌کنیم. در نتیجه پنجره‌ای همانند شکل بعد باز می‌شود.



شکل ۱۹. پنجره اضافه کردن فایل به پروژه

در شکل بالا از قاب بالایی نوع فایل را انتخاب کرده (در اینجا فایل با پسوند ".c") و سپس در قاب Name، نام فایل و در قاب Location، محل ذخیره شدن فایل را مشخص می‌کنیم. در انتها بر روی کلید Add می‌زنیم. سپس در فایل ایجاد شده، برنامه را به صورت زیر می‌نویسیم:

```
#include "lpc_gpio.h"
#include "lpc_pinsel.h"

void Delay (uint32_t Dly){
    Dly*=20000;
    for(; Dly>0; Dly--);
}

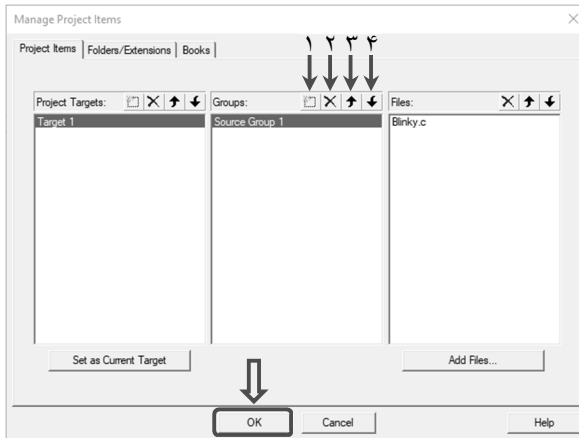
int main(void){
    PINSEL_ConfigPin(0,2,0);           // P0.2 (LED4)
    PINSEL_ConfigPin(0,3,0);           // P0.3 (LED3)
    GPIO_SetDir(0, (1<<2), 1);
    GPIO_SetDir(0, (1<<3), 1);

    while(1){
        GPIO_ClearValue(0, (1<<2));    // LED4 ON
        GPIO_SetValue(0, (1<<3));      // LED3 Off
        Delay(500);                    // Delay 500ms
        GPIO_SetValue(0, (1<<2));      // LED4 Off
        GPIO_ClearValue(0, (1<<3));    // LED3 ON
        Delay(500);                    // Delay 500ms
    }
}
```

پس از نوشتن برنامه، از منوی File گزینه Save (Ctrl+S) را انتخاب کرده و برنامه را ذخیره می‌کنیم. در ادامه برای کارکرد صحیح برنامه باید چند فایل به نمودار درختی برنامه اضافه شود. این فایل‌ها عبارتند از:

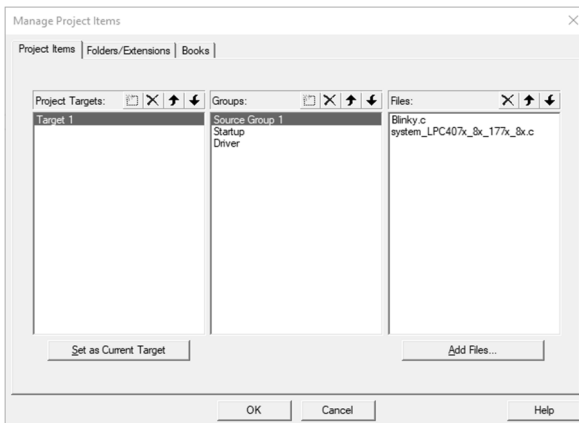
- فایل "startup_LPC407x_8x_177x_8x.s" در آدرس "Example\Core\Device\NXP\LPC407x_8x_177x_8x\Source\Templates\ARM"
 - فایل "system_LPC407x_8x_177x_8x.c" در آدرس "Example\Core\Device\NXP\LPC407x_8x_177x_8x\Source\Templates"
 - فایل‌های "lpc_gpio.c"، "lpc_pinsel.c" و "lpc_clkpwr.c" در آدرس "Example\Drivers\source"
- تذکر: اگر فایل‌های موجود در DVD را به صورت ذکر شده در زیر، به برنامه خود اضافه کنید، پس از درآوردن DVD از رایانه، دیگر برنامه شما ترجمه نخواهد شد. بنابراین ابتدا از DVD کتاب و در پوشه Example، هر دو پوشه Core و Drivers را کپی و در کنار پوشه پروژه خود بچسبانید. سپس فایل‌های موجود در آنها را به صورت ذکر شده در زیر، به برنامه خود اضافه کنید.

برای اضافه کردن این پنج فایل ابتدا از منوی Project، گزینه Manage و سپس گزینه "Project Items" را انتخاب



کنید. در نتیجه پنجره‌ای همانند شکل روبرو باز می‌شود.

شکل ۲۰. پنجره مدیریت پروژه

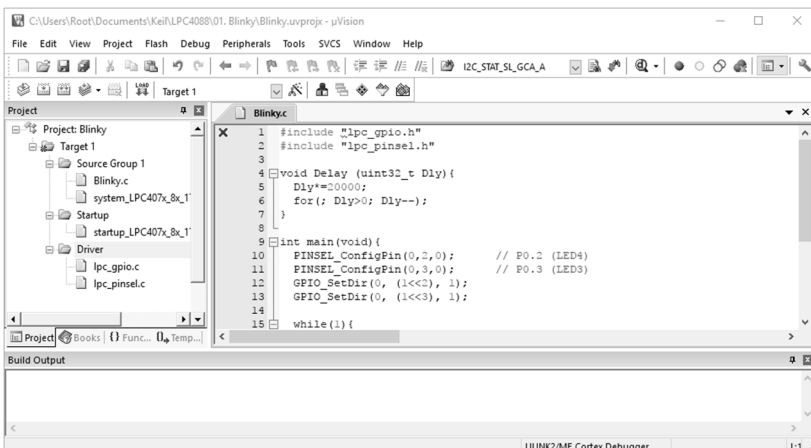


در شکل بالا و در ستون Groups با زدن کلید (۱)، خط جدیدی ایجاد شده که در آن کلمه Startup را می‌نویسیم. با زدن دوباره کلید (۱)، خط جدید دیگری ایجاد شده که در آن کلمه Driver را می‌نویسیم. بدین ترتیب در نمودار درختی برنامه، دو پوشه با نام‌های Startup و Driver به پروژه اضافه می‌شود. سپس پوشه "Source Group 1" را انتخاب کرده و در انتهای ستون Files، کلید "Add Files" را می‌زنیم که در پنجره "Add Files to Group" به آدرس Example\Core\Device\NXP\LPC407x "_8x_177x_8x\Source\Templates (در همان دو پوشه‌ای که کنار پروژه خود چسبانده‌اید)

شکل ۲۱. پنجره مدیریت پروژه

رفته و فایل "system_LPC407x_8x_177x_8x.c" را انتخاب و با زدن کلید Add، آنرا به این پوشه اضافه و در انتها کلید Close را می‌زنیم. در این هنگام باید پنجره مدیریت پروژه به صورت شکل بالا باشد. چهار فایل دیگر را نیز همانند این فایل به پروژه اضافه کنید.

تذکر ۱: فایل "startup_LPC407x_8x_177x_8x.s" را به پوشه Startup و فایل‌های "lpc_gpio.c"، "lpc_pinsel.c" و "lpc_clkpwr.c" را به پوشه Driver اضافه کنید.

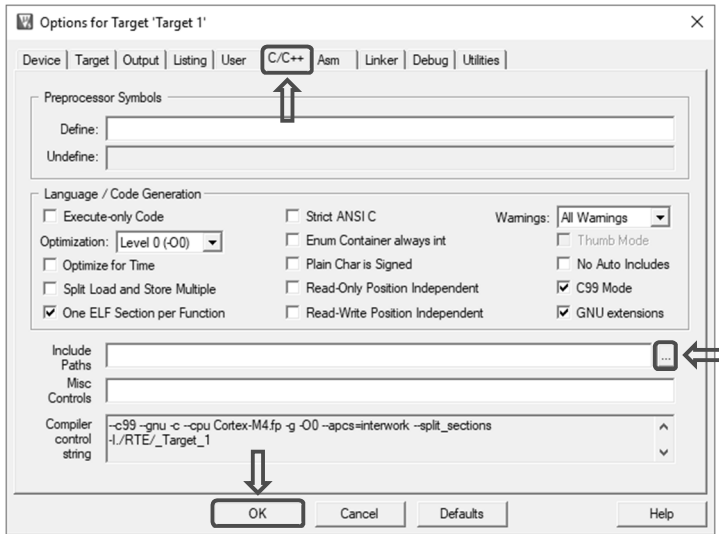


تذکر ۲: همه فایل‌ها را می‌توان در یک پوشه قرار داد، اما این کار در پروژه‌های بزرگ، مدیریت پروژه را سخت‌تر می‌کند. با پوشه‌بندی پروژه، مدیریت آن آسان‌تر می‌شود.

در انتها در پنجره مدیریت پروژه، کلید OK را بزنید. در این صورت در نرم‌افزار کیل باید شکل روبرو را داشته باشید.

شکل ۲۲. نمایشی از پروژه در نرم‌افزار کیل

در خط اول برنامه (در ویرایشگر) ضربدر قرمزی را مشاهده می‌کنید که نشان دهنده وجود خطا در برنامه است. این خطا به خاطر معرفی نکردن فایل‌های سرآیند (فایل‌هایی با پسوند ".h") به برنامه است. برای این کار باید از منوی Project گزینه "Options for Target" را بزنیم (Alt+F7) که در نتیجه پنجره "Options for Target" باز می‌شود. در این پنجره همچون شکل زیر، سربرگ "C/C++" را انتخاب می‌کنیم.

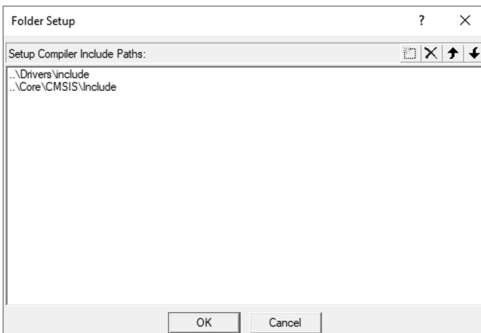


شکل ۲۳. سربرگ C/C++ در پنجره "Options for Target"



در این سربرگ کلید "... " را می‌زنیم. در نتیجه پنجره‌ای مانند شکل روبرو باز می‌شود.

شکل ۲۴. پنجره "Folder Setup"



در این پنجره بر روی کلید مشخص شده می‌زنیم. در این حالت خطی به پنجره اضافه شده که در سمت راست آن کلید "... " وجود دارد. با زدن این کلید پنجره‌ای باز شده که در این پنجره به آدرس "Example\Drivers\include" رفته و سپس کلید "Select Folder" را می‌زنیم. دوباره همین کار را برای آدرس "Example\Core\CMSIS\Include" تکرار می‌کنیم. در نهایت باید شکل روبرو را داشته باشیم.

شکل ۲۵. پنجره "Folder Setup"

در پنجره شکل بالا بر روی کلید OK زده و در پنجره "Options for Target" به سربرگ Output رفته و گزینه "Create HEX File" را علامت‌دار می‌کنیم. سپس بر روی کلید OK می‌زنیم. در نهایت باید آن علامت ضربدر قرمز

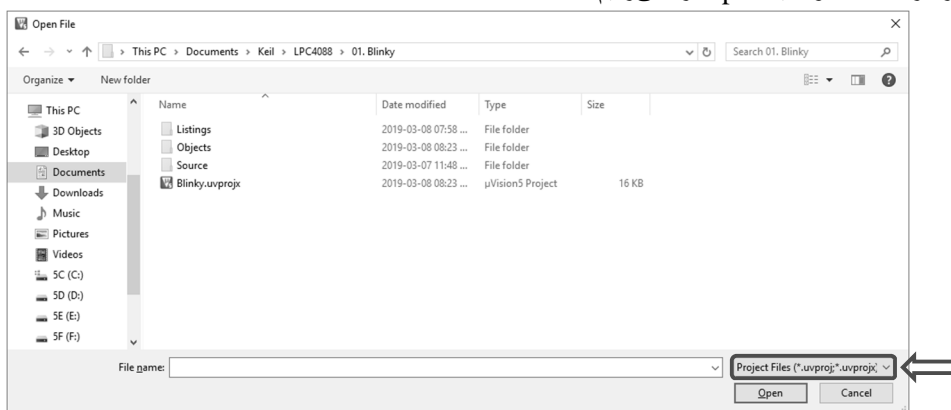
رنگ حذف شده باشد. نوشتن برنامه در این مرحله کامل شده و برنامه آماده ترجمه می‌باشد.
تذکر: برای کارکرد صحیح برنامه پس از انتقال به برد مدار چاپی باید در پنجره "Options for Target" و در سربرگ Target، مقدار نوسانگر وصل شده به تراشه را در قاب "Xtal (MHz)" تنظیم کنیم. در این سربرگ تنظیمات حافظه متناسب با نیاز برنامه باید انجام شود که در برنامه چشمکزن، بدلیل سادگی، نیازی به تنظیمات خاصی برای حافظه نیست.

۵/۳/۲ بازکردن یک پروژه از قبل ایجادشده

یک پروژه را به پنج روش می‌توانیم باز کنیم.

1. انتخاب گزینه Open از منوی File
2. زدن کلیدهای ترکیبی "Ctrl+O" از صفحه کلید
3. انتخاب ابزار Open از نوار ابزار File
4. انتخاب گزینه "Open Project" از منوی Project
5. رفتن به آدرس پروژه و بازکردن فایل اصلی پروژه با پسوند ".uvproj" یا ".uvprojx".

پس از انجام سه مورد اول با پنجره شکل زیر روبرو می‌شویم. در این پنجره ابتدا باید به آدرس پروژه مورد نظر رفته و در قاب مشخص شده در شکل زیر نوع فایل را در منوی کشویی آن، از نوع پروژه انتخاب کنیم. سپس فایل پروژه را انتخاب و کلید Open را می‌زنیم.



شکل ۲۶. پنجره "Open File"

تذکر: اگر پروژه‌ای توسط چهار مورد اول باز شود، پروژه قبلی بسته شده و پروژه جدید جای آن باز می‌شود و اگر توسط پنجمین مورد باز شود، پروژه قبلی همچنان باز بوده و پروژه جدید در کنار آن باز می‌شود.

۵/۳/۳ شبیه‌سازی برنامه در نرم‌افزار کیل

نرم‌افزار کیل دارای قابلیت شبیه‌سازی برنامه برای همه میکروهای سری ARM7، ARM9 و Cortex-M است. با استفاده از کیل می‌توان برنامه‌های نوشته شده برای تراشه‌هایی با هسته‌های زیر را شبیه‌سازی کرد:

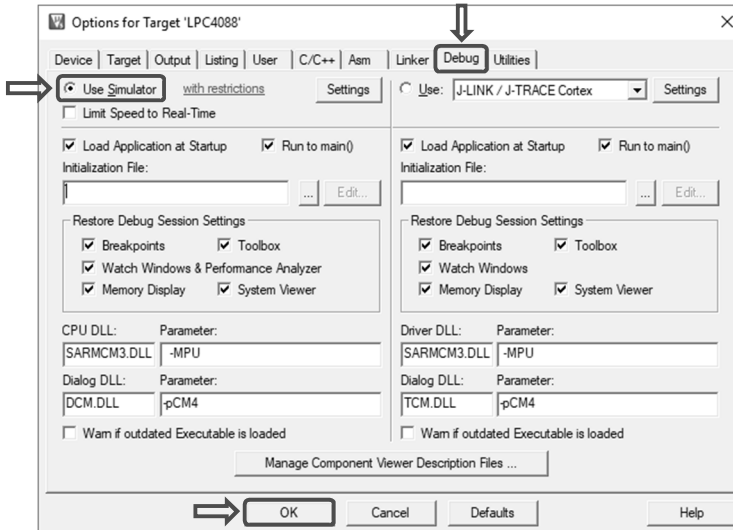
- هسته‌های ARM7 و ARM9
- هسته‌های Cortex-M0 و Cortex-M0+
- هسته Cortex-M3
- هسته‌های Cortex-M4 و Cortex-M4P
- هسته Cortex-M7 از طریق FVM^۱
- هسته‌های امن SC000 و SC300

تذکر: برای شبیه‌سازی هسته‌های قدیمی مانند ARM7 و ARM9 می‌بایست فایل "MDK79525.EXE" در DVD کتاب و از آدرس "Compiler\Keil\Software Pack" نصب شود.

شبیه‌سازی برنامه سبب می‌شود که در هنگام ساخت برد مدار چاپی (زمانی که هنوز برد به صورت کامل در دسترس

نیست) بتوانیم برنامه را خطیابی کرده و عیب و ایرادهای آن را برطرف کنیم.

برای استفاده از حالت شبیه‌سازی در نرم‌افزار کایل، ابتدا برنامه باید ترجمه شده و سپس این حالت فعال شود. برای فعال کردن این حالت، از منوی Project گزینه "Options for Target" را انتخاب می‌کنیم (Alt+F7). در پنجره "Options for Target" سربرگ Debug را مطابق شکل زیر انتخاب و در سمت چپ پنجره، گزینه رادیویی "Use Simulator" را انتخاب می‌کنیم. سپس کلید OK را می‌زنیم.



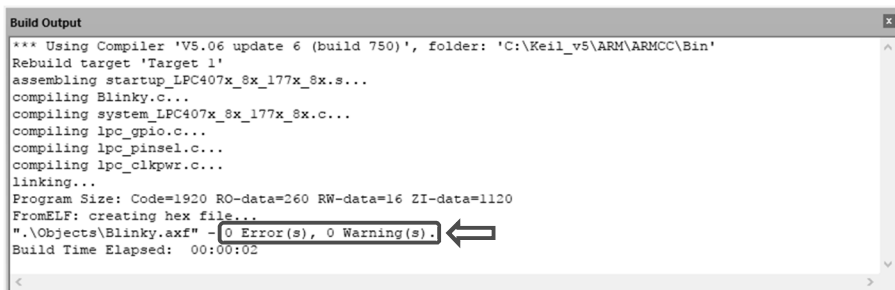
شکل ۲۷. سربرگ Debug از پنجره "Options for Target"

تذکر: در پنجره شکل بالا، سمت راست پنجره مربوط به خطیابی از طریق سخت‌افزار بوده و سمت چپ پنجره مربوط به خطیابی از طریق نرم‌افزار (خود نرم‌افزار کایل) می‌باشد. برای شروع شبیه‌سازی باید از منوی Debug گزینه "Start/Stop Debug Session" انتخاب شود (Ctrl+F5).

۵/۴ چگونگی ترجمه (Compile) پروژه و انتقال آن به برد مدار چاپی

۵/۴/۱ چگونگی ترجمه پروژه

برای ترجمه پروژه، از منوی Project گزینه "Rebuild all Target Files" را انتخاب کنید یا در نوار ابزار Build دکمه Rebuild را بزنید. در این هنگام فرآیند ترجمه آغاز شده و در نهایت باید پنجره "Build Output" را شبیه شکل زیر مشاهده نمایید.



شکل ۲۸. پنجره "Build Output"

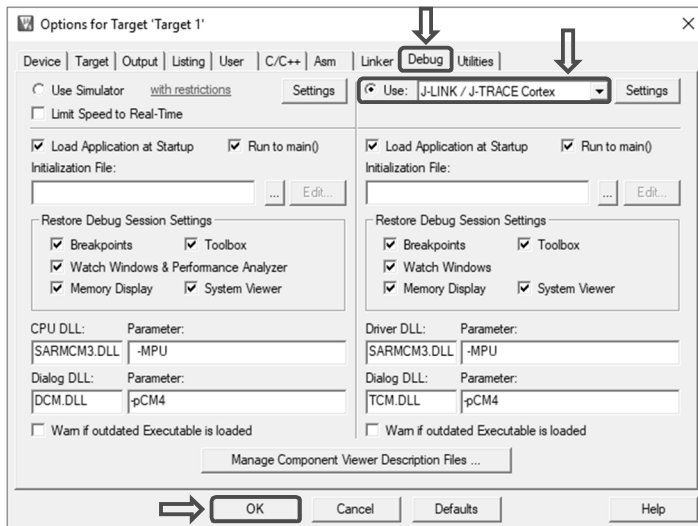
پس از ترجمه، تعداد خطاها و اخطارها در پنجره "Build Output" نشان داده می‌شود. اگر ترجمه دارای اخطار باشد، ترجمه انجام شده و برنامه کار خواهد کرد اما اگر ترجمه دارای خطا باشد، ترجمه متناسب با نوع خطا، انجام

نشده یا در صورت انجام شدن، برنامه کار نخواهد کرد. همانند شکل بالا، ترجمه نباید دارای هیچ خطایی باشد و تعداد خطاها طبق شکل بالا باید ۰ باشد. تذکر: در صورت مشاهده نکردن پنجره بالا، از منوی View گزینه "Build Output Window" را انتخاب نمایید.

۵/۴/۲ انتقال پروژه به برد مدار چاپی

برای انتقال فایل hex حاصل از ترجمه برنامه به برد مدار چاپی، ابتدا باید تنظیمات مربوط به برنامه‌ریز انجام شود. بنابراین با انتخاب گزینه "Options for Target" از منوی Project (Alt+F7) پنجره "Options for Target" باز شده و با انتخاب سربرگ Debug، پنجره همانند شکل روبرو خواهد شد.

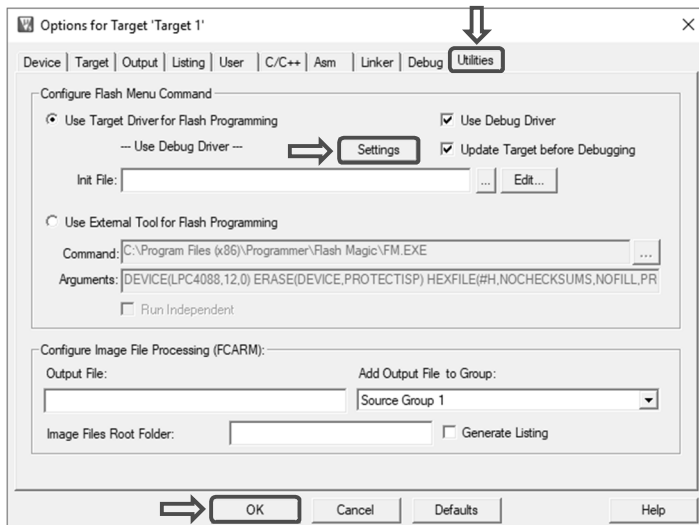
شکل ۲۹. سربرگ Debug در پنجره "Options for Target"



سپس در پنجره شکل قبل، گزینه رادیویی سمت راست پنجره را انتخاب و در منوی کشویی این گزینه رادیویی، نام برنامه‌ریز را مشخص می‌کنیم.

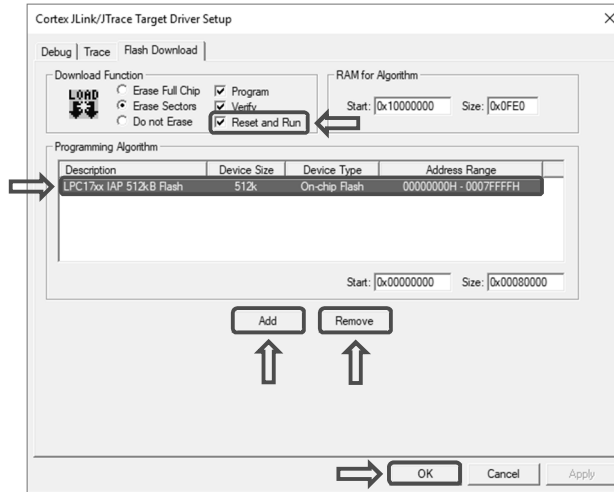
تذکر: در شکل قبل، گزینه رادیویی سمت راست، به صورت پیش‌فرض در حالت انتخاب شده قرار دارد. سپس همانند شکل زیر به سربرگ Utilities می‌رویم. در قسمت "Configure Flash Menu Command" دو گزینه رادیویی وجود دارد که اولی برای انتقال فایل hex از طریق سخت‌افزار خارجی (برنامه‌ریز) و دومی از طریق نرم‌افزار جانبی (مانند نرم‌افزار "Flash Magic" به همراه مبدل USB به سریال) استفاده می‌شود.

تذکر: در صورت استفاده از گزینه رادیویی دوم، متناسب با نرم‌افزار جانبی استفاده شده، ممکن است نتوانید برنامه را خطاییابی کنید. مثلاً اگر از نرم‌افزار "Flash Magic" به عنوان نرم‌افزار جانبی استفاده کنید (همانطور که در شکل زیر مشخص است) دیگر برنامه نوشته شده را نمی‌توانید خطاییابی کنید زیرا نرم‌افزار "Flash Magic" فاقد این قابلیت است.



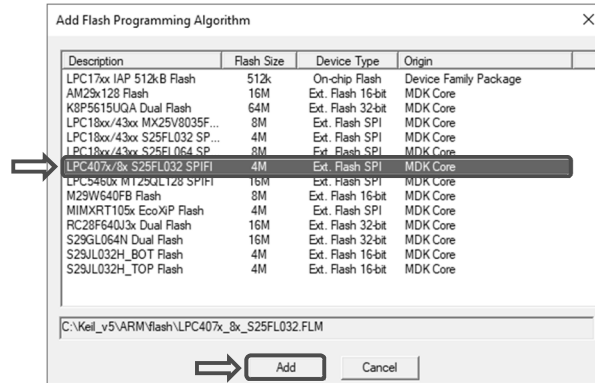
شکل ۳۰. سربرگ Utilities در پنجره "Options for Target"

در شکل بالا و در قسمت "Configure Flash Menu Command" بر روی کلید Settings بزنید. در نتیجه پنجره‌ای همانند شکل زیر باز می‌شود.



شکل ۳۱. پنجره انتخاب تراشه برای برنامه‌ریزی

در پنجره شکل قبل در قسمت "Download Function" جمعیه بررسی "Reset and Run" را انتخاب کنید. سپس باید نوع تراشه استفاده شده را برای برنامه‌ریزی، انتخاب کنیم. برای این کار ابتدا تراشه موجود در قسمت "Programming Algorithm" را انتخاب و سپس بر روی کلید Remove (در پایین آن) می‌زنیم. سپس بر روی کلید Add می‌زنیم. در نتیجه پنجره‌ای همانند شکل زیر باز می‌شود.



شکل ۳۲. پنجره "Add Flash Programming Algorithm"

در این پنجره نوع تراشه موجود روی برد را انتخاب و سپس کلید Add را می‌زنیم. سپس در پنجره انتخاب تراشه برای برنامه‌ریزی، کلید OK را زده و بعد از آن در سربرگ Utilities نیز کلید OK را می‌زنیم. پس از تنظیم برنامه و هنگامی که برنامه‌ریزی به رایانه وصل است، از منوی Flash گزینه Download (F8) را انتخاب می‌کنیم. سپس نرم‌افزار کیل از طریق برنامه‌ریزی با برد مدار چاپی ارتباط برقرار کرده و فایل hex را به تراشه روی برد منتقل می‌کند.

فصل

۲۲

خطایابی و ردیابی

واژه‌های کلیدی:

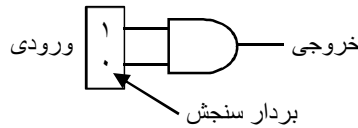
TAP: Test Access Port

IEEE: Institute of Electrical and Electronics Engineers

درگاه دسترسی به JTAG
انجمن مهندسان برق و الکترونیک

۲۲/۱ تاریخچه خطایابی

همزمان با تولید تراشه‌ها، نیاز به بررسی عملکرد آنها نیز بوجود آمد. هر قطعه/بخش از تراشه دارای تعدادی ورودی و خروجی بوده که هر حالتی در ورودی‌های آن، یک بردار سنجش محسوب می‌شود.



شکل ۱. بردار سنجش

هر کدام از بردارهای سنجش به قطعه/بخش مربوطه اعمال شده و خروجی ناشی از آن با الگوهای پیش‌بینی شده مقایسه می‌شود. این کار برای همه بردارهای سنجش به صورت پشت‌سرم انجام شده و در صورت نبودن تفاوت در هیچ‌کدام از مقایسه‌ها، آن قطعه/بخش بدون مشکل خواهد بود. تعداد بردارهای سنجش برای هر قطعه/بخش از رابطه $Q = 2^{(x+y)}$ محاسبه می‌شود که Q برابر با حداقل تعداد بردارهای سنجش، X برابر با تعداد ورودی و Y برابر با تعداد قطعه/بخش دارای حافظه است.

در شکل بالا به دلیل اینکه قطعه AND دارای قطعه/بخش حافظه دار نیست، بنابراین دارای $Q = 2^2 = 4$ بردار سنجش است. در مداری با ۲۵ ورودی و ۵۰ قطعه/بخش حافظه دار، Q برابر با $3/8 \times 10^{22} \approx 2^{75}$ حالت خواهد شد و اگر پردازنده با سرعت 1GHz کار کند، برای بررسی همه این حالت‌ها تقریباً به ۱/۲ میلیون سال نیاز است. در سال‌های ۱۹۷۰ مهندسان با چنین مشکلاتی مواجه‌بودند و این زمان‌های طولانی، امکان توسعه و خطایابی مدارها را بسیار سخت و یا غیرممکن می‌ساخت. در اوایل سال‌های ۱۹۷۰، شرکت IBM، روش LSSD را برای رفع این مشکل ابداع کرد. در این روش چهار پایه (یک ورودی، یک خروجی و دو پایه ضربان) به تراشه اضافه شد.

در ابتدای سال‌های ۱۹۸۰، مشکل جدیدی مطرح شد که "افزایش پیچیدگی PCB‌هایی با فشردگی و تعداد قطعات بیشتر" در سطح برد بود. گروه JETAG (تشکیل شده از بزرگترین تولیدکنندگان تراشه در اروپا) اولین گروهی بود که در سال ۱۹۸۵ برای رفع این مشکل تشکیل شد. در سال ۱۹۸۶ با اضافه‌شدن چند شرکت آمریکایی به این گروه، نام آن به JTAG تغییر یافت. گروه JTAG برای حل این مشکل بعد از ۵ سال بحث و بررسی در نهایت معماری را پیشنهاد داد که بسیار به روش LSSD شبیه بود. تفاوت این دو روش در وجود ذخیره‌گاه‌ها در مرز تراشه (متصل به پایه‌های تراشه) بود که توسط زنجیره‌ای با هم در ارتباط بودند (شکل ۲۰۳). به همین دلیل به این روش "بررسی مرزی" می‌گویند. این روش طبق استاندارد "درگاه دسترسی به JTAG (TAP) و بررسی مرزی" توسط سازمان IEEE در سال ۱۹۹۰ بنام "استاندارد IEEE به شماره ۱۱۴۹/۱ (JTAG)" استاندارد سازی شد.

۲۲/۲ ویژگی‌های خطایابی در تراشه‌های LPC408x/7x

- حمایت از حالت‌های JTAG استاندارد و خطایابی سیمی سریالی (SWD)
- دسترسی مستقیم از طریق خطایابی به تمامی حافظه‌ها، ثبات‌ها و اجزای جانبی
- لازم نداشتن تراشه/دستگاه به هیچ تجهیزات اضافی برای خطایابی
- قابلیت ردیابی دستورهای در حال اجرا در پردازنده توسط درگاه ردیابی (Trace)
- دارای هشت نقطه توقف. شش نقطه توقف دستور که می‌توانند با ترسیم دوباره آدرس‌های دستور در تصحیح کد استفاده شوند و دو مقایسه‌گر داده که می‌توانند با ترسیم دوباره آدرس‌ها، برای درست‌کردن داده‌ها با مقدارهای دقیق، استفاده شوند.
- دارای چهار نقطه نگهبان داده که می‌توان از آن‌ها برای تحریک بخش ردیابی استفاده کرد.
- قابلیت ردیابی نرم‌افزاری که امکان ردیابی کنترل شده بیشتری را از طریق برنامه کاربر فراهم می‌کند.

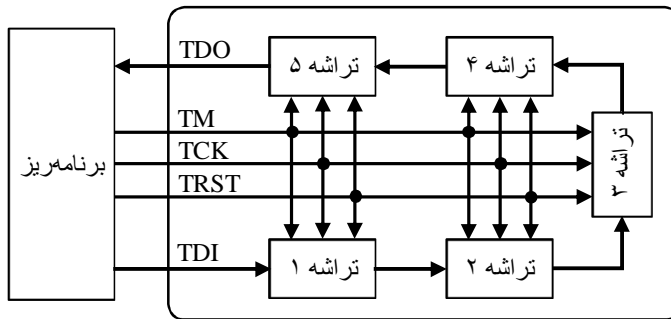
۲۲/۳ توضیحاتی در مورد خطایابی و ردیابی

هسته Cortex-M4 دارای قابلیت خطایابی و ردیابی است. این هسته علاوه بر خطایابی با استاندارد JTAG و ردیابی موازی، از خطایابی SWD نیز حمایت می‌کند. زمانی که خطایابی در تراشه‌های LPC408x/7x شروع شود، بخش خطایابی به صورت پیش‌فرض وارد حالت JTAG خواهد شد. تراشه به راحتی می‌تواند توسط برنامه‌ریز از حالت JTAG به حالت SWD برود.

حالت ردیابی از یک رابط موازی ۴-بیتی (با استفاده از ۵ پایه) حمایت می‌کند. حالت ردیابی در Cortex-M4 نسبت به خانواده ARM7 متفاوت بوده و در آن از ۵ پایه بجای ۱۰ پایه استفاده می‌شود. Cortex-M4 از هشت نقطه توقف و چهار نقطه نگهبان حمایت می‌کند.

در زمان خطایابی باید محدودیت‌های تراشه مانند بیدار نشدن تراشه از حالت‌های خواب عمیق و کاهش مصرف توان به روش معمول بخاطر ترکیب شدن بخش خطایابی با هسته Cortex-M4، در نظر گرفته شود. توصیه می‌شود که از این حالت‌ها در طی خطایابی استفاده نشود. برخی از این محدودیت‌ها به صورت زیر می‌باشند:

- زمانی که برنامه از طریق رابط JTAG/SWD به تراشه ارسال شد، برنامه‌ریز باید از برد موردنظر جدا شود. پس از آن تراشه بکار افتاده و این مسئله امکان بیداری از حالت‌های خواب عمیق و کاهش مصرف توان را فراهم می‌سازد.
- حالت‌های کاهش مصرف توان در هنگام خطایابی با حالت معمول آنها متفاوت هستند. این تفاوت‌ها بدان معناست که ارزیابی مصرف توان نباید در زمان خطایابی برای حالت‌های کاهش مصرف توان انجام شود، زیرا نتایج به‌دست‌آمده از ارزیابی مصرف توان در حالت‌های کاهش مصرف توان نسبت به زمانی که خطایابی انجام نمی‌شود بالاتر خواهد بود.
- با توقف پردازنده در هنگام خطایابی، زمان‌سنج ثانیه‌ای نیز به‌صورت خودکار متوقف شده ولی دیگر اجزای جانبی تحت تأثیر قرار نمی‌گیرند.
- اگر حالت محافظت از خواندن کد فعال شده باشد، خطایابی غیرفعال می‌شود.



شکل ۲. حلقه خطایابی در یک برد مدار چاپی

حلقه خطایابی عبارت است از مسیری در برد مدار چاپی که از TDI شروع و به TDO ختم می‌شود. حداکثر بسامد کاری JTAG و پایه TCK، به تمام تراشه‌های موجود در این حلقه بستگی داشته و با سرعت کندترین تراشه موجود در این حلقه برابر است و معمولاً بین 10MHz تا 100MHz می‌باشد. این مقدار برای تراشه‌های LPC408x/7x به‌صورت پیش‌فرض 10MHz است (بخش ۲۲/۵/۴).

۲۲/۴ توضیحاتی در مورد پایه‌های خطایابی و ردیابی

جدول زیر، کاربرد پایه‌های مربوط به حالت خطایابی و حالت ردیابی را نشان می‌دهد. بعضی از پایه‌های این حالت‌ها با کاربردهای دیگر مشترک‌اند بنابراین نمی‌توان از این حالت‌ها به‌صورت هم‌زمان استفاده کرد. استفاده از حالت ردیابی نیازمند ۵ پایه است که برخی از این پایه‌ها می‌توانند توسط برنامه‌کاربر استفاده شوند.

جدول ۱. توضیحات مربوط به پایه‌های حالت JTAG، حالت SWD و حالت ردیابی

نام پایه	نوع	توضیحات
JTAG_TCK	پایه‌های حالت JTAG	ورودی JTAG و ضربان JTAG. این پایه زمانی که تراشه در حالت JTAG قرار دارد، تأمین‌کننده ضربان بخش خطایابی است. ضربان دریافتی از این پایه باید ۶ برابر کمتر از ضربان پردازنده (CCLK) باشد.

JTAG_TMS	ورودی	انتخاب حالت JTAG. از مقدار این پایه در لبه بالارونده JTAG_TCK، برای تعیین حالت بعدی کنترل‌کننده استفاده می‌شود. این پایه دارای یک بالاکش درونی مطابق با استاندارد JTAG است.
JTAG_TDI	ورودی	ورودی داده JTAG. داده/دستور از طریق این پایه و از برنامه‌ریز ^(۱) دریافت می‌شود. این پایه دارای یک بالاکش درونی مطابق با استاندارد JTAG است.
JTAG_TDO	خروجی	خروجی داده JTAG. داده از طریق این پایه و روی لبه پایین‌رونده موج JTAG_TCK به برنامه‌ریز ارسال می‌شود.
JTAG_TRST	ورودی	راه‌اندازی مجدد JTAG. از این پایه می‌توان برای راه‌اندازی مجدد بخش خطیابی/ردیابی استفاده کرد. این پایه دارای یک بالاکش درونی مطابق با استاندارد JTAG است.

پایه‌های حالت SWD

SWDCLK	ورودی	ضربان SWD. این پایه زمانی که تراشه در حالت SWD قرار دارد، تأمین‌کننده ضربان بخش خطیابی است. این پایه با پایه JTAG_TCK از نظر کاربردی مشترک است. ضربان دریافتی از این پایه باید ۶ برابر کمتر از ضربان پردازنده (CCLK) باشد.
SWDIO	ورودی/خروجی	ورودی/خروجی داده SWD. برنامه‌ریز از این پایه برای برقراری ارتباط و کنترل پردازنده Cortex-M4 استفاده می‌کند. این پایه با پایه JTAG_TMS از نظر کاربردی مشترک است.
SWO	خروجی	خروجی SWD. این پایه به‌صورت انتخابی داده‌ها را برای ارزیابی به برنامه‌ریز می‌فرستد. این پایه با پایه JTAG_TDO از نظر کاربردی مشترک است.

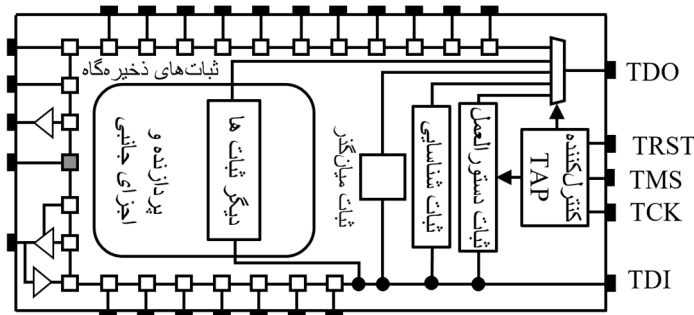
پایه‌های حالت ردیابی

TRACECLK	ورودی	ضربان ردیابی. این پایه زمانی که تراشه در حالت ردیابی قرار دارد، تأمین‌کننده ضربان بخش ردیابی است.
TRACEDAT	خروجی	داده ردیابی (بیت‌های ۰ تا ۳). داده حالت ردیابی ETM به صورت فشرده‌شده از طریق این پایه‌ها به برنامه‌ریز ارسال می‌شود.

Programmer [۱]

۲۲/۵ خطیابی از طریق JTAG

در شکل زیر معماری پیشنهاد شده توسط گروه JTAG را مشاهده می‌کنید.



شکل ۳. معماری استاندارد IEEE به شماره ۱۱۴۹/۱ (JTAG)

استاندارد JTAG، معماری خطیابی JTAG بوده و شامل موارد زیر است:

- مجموعه‌ای از چهار پایه اختصاصی (TMS، TCK، TDO و TRST) و پایه انتخابی^۱ TRST
- یک ثبت ذخیره‌گاه، یک ثبت دستور و چند ثبت داده (شامل ثبت میان‌گزر ۱-بیتی و ثبت شناسایی ۳۲-بیتی)
- یک کنترل‌کننده TAP

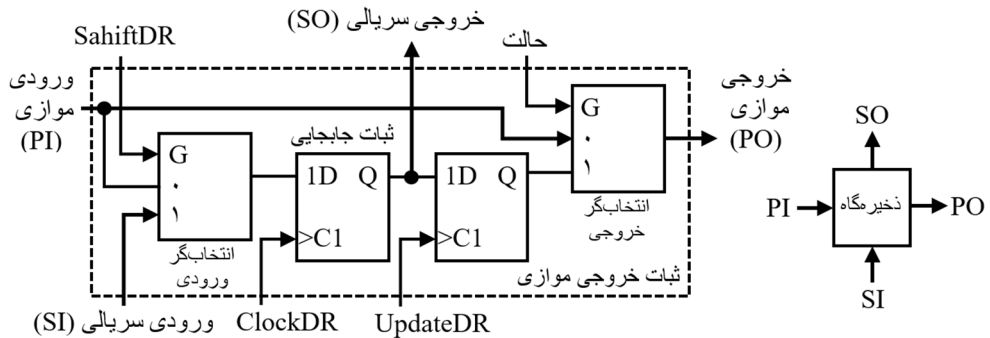
۲۲/۵/۱ ذخیره‌گاه

در استاندارد JTAG، به هر پایه (غیر از پایه‌هایی مانند تغذیه و آنالوگ)، نگهدارنده چند منظوره‌ای با نام ذخیره‌گاه اضافه شده است. تعدادی از ذخیره‌گاه‌ها بین پردازنده و پایه‌های ورودی، خروجی، خط‌های کنترل پایه‌های دو جهته و پایه‌های سه حالت قرار گرفته و تعدادی دیگر از آنها در کنار بخش‌هایی مانند ADC، DAC، PLL و غیره قرار گرفته اند. همان طور که در شکل زیر مشاهده می‌کنید، هر ذخیره‌گاه دارای چندین خط ورودی، خروجی، ضربان و کنترل است. همچنین هر ذخیره‌گاه دارای یک مبدل جابجایی بوده که می‌تواند داده‌ها را بین ورودی موازی و خروجی موازی جابجا کند که با

^۱ پایه‌هایی هستند که JTAG بدون آنها هم قابلیت کارکردن را دارد، برعکس پایه‌های اختصاصی که وجود آنها ضروری است.

این کار می‌تواند بین پایه تراشه و پردازنده ارتباط برقرار کند. همچنین می‌تواند داده‌ها را بین ورودی سریال و خروجی سریال نیز جابجا کند که با این کار می‌تواند بین ذخیره‌گاه قبلی و ذخیره‌گاه بعدی (در زنجیره) ارتباط برقرار کند. ذخیره‌گاه‌ها با هم در ارتباط بوده و تشکیل زنجیره‌ای از ذخیره‌گاه‌ها را می‌دهند که از پایه TDI شروع و به پایه TDO ختم می‌شوند. به این زنجیره که به صورت سری به هم متصل هستند "ثبات ذخیره‌گاه" می‌گویند. همه بخش‌های دیجیتال تراشه در داخل این زنجیره قرار گرفته و بخش‌های آنالوگ می‌توانند خارج از آن قرار بگیرند. پایه‌ای که فقط دارای قابلیت آنالوگ باشد، فاقد ذخیره‌گاه بوده ولی اگر در کنار قابلیت آنالوگ دارای قابلیت دیجیتال نیز باشد، به دلیل قابلیت دیجیتال دارای ذخیره‌گاه نیز خواهد بود. هر پایه ورودی/خروجی حداقل دارای دو ذخیره‌گاه (یکی برای قسمت ورودی و یکی برای قسمت خروجی) است. ذخیره‌گاه قسمت ورودی، داده را از پایه تراشه و از طریق PI دریافت و آنرا از طریق PO به پردازنده ارسال می‌کند. ذخیره‌گاه قسمت خروجی، داده را از پردازنده و از طریق PI دریافت و آنرا از طریق PO به پایه تراشه ارسال می‌کند. هر ذخیره‌گاه می‌تواند:

- داده را از PI به PO منتقل کند (بین پردازنده و پایه تراشه).
- از طریق خروجی سریالی (SO) خود، داده را به ورودی سریالی (SI) ذخیره‌گاه بعدی منتقل و از طریق SI خود، داده را از SO ذخیره‌گاه قبلی دریافت کند.



شکل ۴. ذخیره‌گاه و نمای داخلی آن

۲۲/۵/۲ فرمان‌های مورد استفاده در خطایابی و ردیابی

هنگامی که فرمانی از سمت رایانه/برنامهریز و از طریق درگاه خطایابی/ردیابی به بخش خطایابی/ردیابی پردازنده ارسال شود، در ثبات دستور قرار می‌گیرد. اگر بیت بالارزش فرمان ارسال شده، ۰ باشد، فرمان ارسال شده، دستور از پیش تعیین شده (دستور) بوده و اگر ۱ باشد، فرمان ارسال شده، دستور استاندارد (یکی از ثبات‌های داده) خواهد بود. این فرمان‌ها در جدول زیر آمده است.

جدول ۲. فرمان‌های مورد استفاده در خطایابی و ردیابی پردازنده‌های ARM

طول ثبات	دستور/ثبات داده	کد فرمان ۸-بیتی	کد فرمان ۴-بیتی
-	EXTST	0b00000000	0b0000
-	SAMPLE	0b00000001	0b0001
-	PRELOAD	0b00000010	0b0010
-	INTEST	0b00000100	0b0100
-	CLAMP	0b00000101	0b0101
-	HIGHZ	0b00000110	0b0110
-	-	0b11110111 تا 0b10000000	-
۳۵	ABORT	0b11111000	0b1000
-	-	0b11111001	0b1001
۳۵	DPACC	0b11111010	0b1010
۳۵	APACC	0b11111011	0b1011
-	-	0b1111110x	0b110x
۳۲	IDCODE	0b11111110	0b1110

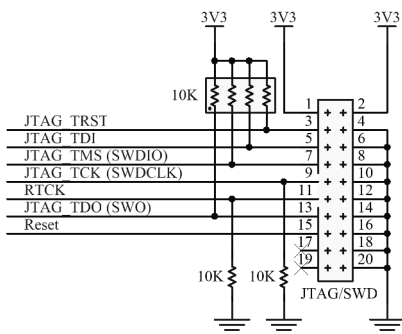
۱	BYPASS	0b11111111	0b1111
---	--------	------------	--------

ثبات‌های بخش خطایابی/ردیابی به دو دسته، ثبات دستور و ثبات‌های داده تقسیم می‌شوند. **ثبات دستور:** ثبات دستور دارای یک بخش جابجایی است که بین پایه‌های TDI و TDO قرار گرفته و دستور را به صورت سری از TDI دریافت، به صورت موازی به قسمت نگهدارنده، ارسال و در آن نگهداری می‌کند. بسته به طول ثبات و تعداد دستورها، بین ثبات دستور و قسمت نگهدارنده می‌تواند تا چند بخش رمزگشایی وجود داشته باشد. کنترل‌کننده TAP، کنترل این ثبات را انجام می‌دهد. این کنترل شامل دریافت سریالی از TDI، ارسال سریالی به TDO و ارسال موازی به قسمت نگهدارنده است. طول ثبات دستور طبق استاندارد JTAG حداقل ۲ بیت بوده و حداکثری برای آن در نظر گرفته نشده است. این طول طبق مستندات شرکت ARM، ثابت بوده و می‌تواند ۴ یا ۸ بیت باشد ولی مقدار آن در پردازنده‌های سری LPC408x/7x برابر با ۵ بیت است.

ثبات‌های داده: ثبات‌های داده طبق استاندارد JTAG شامل ثبات‌های میان‌گذر و شناسایی بوده و در پردازنده‌های ARM، شامل ثبات‌های میان‌گذر، شناسایی، عدم موفقیت، DPACC و APACC می‌باشند. ثبات‌های داده عبارتند از:

- **ثبات میان‌گذر:** ثبات میان‌گذر، ثباتی ۱-بیتی بوده، توسط فرمان BYPASS انتخاب شده و برای جابجایی فرمان/داده از TDI به TDO استفاده می‌شود. این ثبات با ایجاد ارتباط مستقیمی از TDI به TDO و فقط با یک تاخیر، راه میان‌بری را برای دورزدن تراشه بوجود می‌آورد. این ثبات دارای ورودی و خروجی موازی نمی‌باشد.
- **ثبات شناسایی:** ثبات شناسایی، ثباتی ۳۲-بیتی بوده، توسط فرمان IDCODE انتخاب شده و برای شناسایی پردازنده متصل به درگاه خطایابی/ردیابی استفاده می‌شود. این ثبات، کد شناسایی ۳۲-بیتی را از تراشه دریافت کرده و به صورت سری و از طریق TDO به خارج از تراشه ارسال می‌کند. کد شناسایی ۳۲-بیتی شامل موارد زیر است:
 - بیت ۰: این بیت همیشه ۱ است.
 - بیت‌های ۱ تا ۱۱: نشان دهنده سازنده تراشه طبق استاندارد JEDEC
 - بیت‌های ۱۲ تا ۲۷: نشان دهنده شماره تراشه
 - بیت‌های ۲۸ تا ۳۱: نشان دهنده نگارش تراشه
- **تذکر:** کد شناسایی برای خانواده LPC408x/7x، مقدار 0x410FC241 است.
- **ثبات توقف:** ثبات توقف، ثباتی ۳۵-بیتی بوده، توسط فرمان ABORT انتخاب شده و برای متوقف کردن ارتباط یا جابجایی فعلی در درگاه خطایابی/ردیابی استفاده می‌شود.
- **ثبات DPACC:** ثبات DPACC، ثباتی ۳۵-بیتی بوده، توسط فرمان DPACC انتخاب شده و برای خواندن و نوشتن در ثبات‌های درگاه خطایابی (DP) استفاده می‌شود.
- **ثبات APACC:** ثبات APACC، ثباتی ۳۵-بیتی بوده، توسط فرمان APACC انتخاب شده و برای خواندن و نوشتن در ثبات‌های درگاه دسترسی (AP) استفاده می‌شود.
- **ثبات ذخیره‌گاه:** ثبات ذخیره‌گاه، ثباتی است که طول آن در تراشه‌های مختلف، متفاوت بوده و برای خواندن مقدار پایه‌های تراشه، خروجی ADC و ورودی DAC داخلی تراشه و غیره استفاده می‌شود.

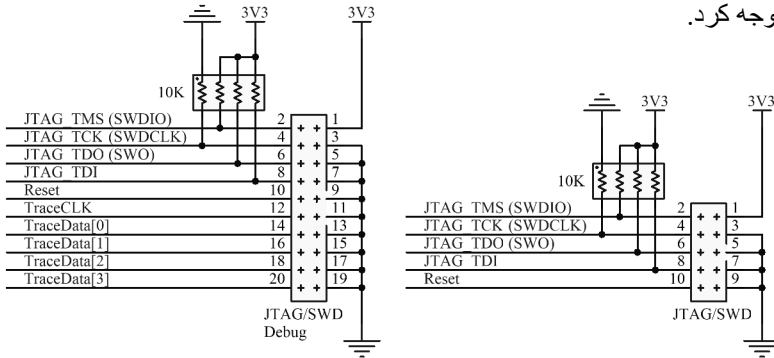
۲۲/۵/۳ کانکتورهای مورد استفاده برای خطایابی و ردیابی



تراشه‌های LPC408x/7x دارای پایه‌های اختصاصی برای JTAG و SWD هستند. برنامه‌ریز و برد خود را می‌توان با کانکتورهای متفاوتی به هم وصل کرد. انتخاب کانکتور خطایابی برای افزودن به یک برد جدید، به حالت خطایابی انتخاب شده در برنامه‌ریز بستگی دارد. برای مثال، ابزارهای خطایابی تراشه‌های ARM از یک کانکتور استاندارد همانند شکل روبرو استفاده می‌کنند. این شکل با تراشه‌های LPC408x/7x مطابقت یافته و پایه‌هایی که به صورت بالاکش در درون تراشه بکار رفته برای آن در نظر گرفته شده‌اند.

شکل ۵. کانکتور ۲۰ پایه استاندارد JTAG در تراشه‌های ARM

ابزارهای خطایابی می‌توانند از یک کانکتور کوچک‌تر، طبق شکل زیر (الف) استفاده کنند. اگر از حالت ردیابی برای خطایابی استفاده شود، می‌توان از کانکتوری استفاده کرد که هر دو حالت ردیابی و خطایابی را همانند شکل زیر (ب) داشته باشد. در هنگام طراحی برد باید به نوع ابزار استفاده‌شده برای خطایابی و همچنین کانکتور موردنیاز آن در طراحی توجه کرد.



الف) کانکتور ۱۰ پایه استاندارد "mini JTAG" (ب) کانکتور ۲۰ پایه استاندارد JTAG

شکل ۶. کانکتورهای استاندارد برای حالت‌های JTAG، SWD و ردیابی

تذکر: هر دو شکل دارای قابلیت SWD هستند و همچنین شکل ب دارای قابلیت ردیابی نیز می‌باشد.

۲۲/۵/۴ فایل‌های BSDL

زبان BSDL، یک زبان قابل‌فهم برای انسان و ماشین بوده و زیرمجموعه‌ای از زبان VHDL است که نحوه پیاده‌سازی استاندارد JTAG را در یک تراشه بیان می‌کند. این زبان در سال ۱۹۹۴ توسط سازمان IEEE معرفی و در سال ۲۰۰۱ اصلاح شد. این زبان دارای ویژگی‌های لازم برای اجرا طبق استاندارد JTAG است. با استفاده از این زبان، ابزارهای مختلف طبق این استاندارد می‌توانند تعداد تراشه‌هایی که خطایابی و ردیابی می‌کنند را گسترش دهند. اگر این ابزارها از چیدمان ذخیره‌گاه‌ها اطلاع داشته باشند، می‌توانند TAP و پایه‌های تراشه را کنترل کنند. هدف از این زبان، ساده‌کردن خطایابی و ردیابی تراشه‌ها و نیز ساده‌کردن گسترش تعداد تراشه‌هایی که ابزارها می‌توانند خطایابی و ردیابی کنند، است.

فایل BSDL با زبان BSDL نوشته شده و شامل موارد زیر می‌باشد (به همراه بخش مرتبط از فایل BSDL تراشه (LPC4088FBD208):

- پایه‌های تراشه و مشخصات آنها

```
port (
p3_12      : inout  bit; -- P3[12]_EMC_D[12]
jtag_tdo_swo : out    bit; -- JTAG_TDO_SWO
p3_3      : inout  bit; -- P3[3]_EMC_D[3]
jtag_tdi   : in     bit; -- JTAG_TDI
```

- نگارش استاندارد مورد استفاده

```
use STD_1149_1_2001.all;
```

- نوع بسته‌بندی، شماره و نام پایه‌های تراشه

```
constant LQFP208 : PIN_MAP_STRING :=
"p3_12      : 1, " &
"jtag_tdo_swo : 2, " &
"p3_3      : 3, " &
"jtag_tdi   : 4, " &
```

- پایه‌های TAP و مشخصات آنها

```
attribute TAP_SCAN_CLOCK of jtag_tck_swclk : signal is (10.00e+06,BOTH);
attribute TAP_SCAN_IN   of jtag_tdi       : signal is true;
attribute TAP_SCAN_MODE of jtag_tms_swdio : signal is true;
attribute TAP_SCAN_OUT  of jtag_tdo_swo   : signal is true;
attribute TAP_SCAN_RESET of jtag_trstn    : signal is true;
```

- دستورهای از پیش تعیین شده و کد آنها

```
"extest      (00000)," &
"sample     (00001)," &
"preload    (00001)," &
"highz     (00010)," &
"clamp     (00011)," &
"idcode     (00100)," &
```

- ثبات‌های داده و کد آنها

```
"bypass     (11111)";
```

- کد شناسایی و مشخصات آن

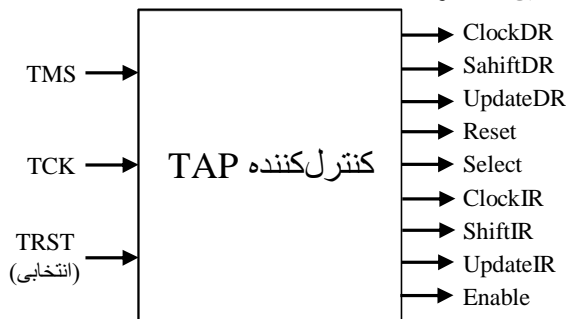
```
attribute IDCODE_REGISTER of LPC408x : entity is
"0xxx"          & -- Version Number
"1111111111000000" & -- Part Number
"00000010101"  & -- Manufacturer ID
"1"            -- Required by IEEE
```

- ثبات ذخیره‌گاه و مشخصات ذخیره‌گاه‌های آن

```
"488 (BC_4, p3_28, INPUT, X          ),"&
"487 (BC_1, p3_28, OUTPUT3, X, 486, 0, Z),"&
"486 (BC_1, *,      CONTROL, 0       ),"&
...
" 2 (BC_4, p3_3, INPUT, X          ),"&
" 1 (BC_1, p3_3, OUTPUT3, X, 0, 0, Z),"&
" 0 (BC_1, *,      CONTROL, 0       );"
```

۲۲/۶ کنترل‌کننده TAP

به مجموع چهار پایه اختصاصی (TMS، TDI، TMS، TCK و TDO) و پایه انتخابی TRST که به صورت ورودی، خروجی و کنترلی هستند، درگاه دسترسی به JTAG (TAP) می‌گویند. پایه TRST در معماری ARM به عنوان یک پایه اختصاصی در نظر گرفته شده، چرا که از آن برای راه‌اندازی مجدد بخش خطایابی استفاده می‌شود. کنترل‌کننده TAP مسئول کنترل کامل بخش خطایابی در تراشه است. کنترل‌کننده، فعال یا غیرفعال شدن ذخیره‌گاه و دریافت یا ارسال داده توسط آن را مشخص می‌کند. با استفاده از کنترل‌کننده می‌توان حافظه‌های فلش و EEPROM داخل تراشه (در صورت وجود) را برنامه‌ریزی کرده و یا در آنها تغییراتی ایجاد کرد. کنترل‌کننده به تغییرات ایجادشده در پایه‌های TMS و TCK پاسخ داده و می‌تواند عملیات‌های پشت‌سرهم را کنترل کند. از مقدار پایه TMS در لبه بالارونده TCK، برای تعیین حالت بعدی کنترل‌کننده استفاده می‌شود. اجرای فرمان دریافت شده، در لبه بالارونده یا لبه پایین‌رونده TCK انجام می‌شود. فرمان ارسال شده به بخش خطایابی/ردیابی پردازنده وارد ثبات دستور شده، سپس رمزگشایی می‌شود. اگر فرمان ارسال شده مربوط به یکی از ثبات‌های JTAG باشد، کنترل‌کننده TAP یکی از ثبات‌های میان‌گنر، شناسایی، دستور و ثبات‌های اختصاصی داخل هسته را انتخاب می‌کند. به صورت همزمان کنترل‌کننده TAP فقط می‌تواند یکی از این ثبات‌ها را برای قرارگیری بین TDI و TDO انتخاب کند.



شکل ۷. کنترل‌کننده TAP

۲۲/۷ تغییر آدرس حافظه در حالت خطایابی

بعد از راه‌اندازی مجدد تراشه، قسمتی از ROM راه‌انداز به آدرس ۰ تغییر آدرس داده و به‌صورت خودکار اجرا خواهد شد. سپس این آدرس به نقطه‌ای از حافظه فلش که کد کاربر از آنجا باید اجرا شود، تغییر می‌کند. در حالت عادی دانستن این تغییر آدرس‌ها توسط کاربر ضروری نیست.

اگرچه خطایابی، اجرای پردازنده را بلافاصله بعد از راه‌اندازی مجدد متوقف می‌کند اما ROM راه‌انداز هنوز شروع آدرس آن از ۰ است و می‌تواند باعث دردرس شود. در حالت ایدال، خطایابی باید آدرس‌دهی را به‌صورت خودکار تصحیح نموده و بنابراین نیازی به انجام آن توسط کاربر نیست.

۲۲/۷/۱ ثبات کنترل تغییر آدرس حافظه (MEMMAP)

این ثبات امکان تغییر آدرس انتهای حافظه‌های ROM راه‌انداز و حافظه فلش را فراهم می‌سازد.

جدول ۳. توضیحاتی در مورد بیت‌های ثبات کنترل تغییر آدرس حافظه (با آدرس 0x400F C040)

بیت	نشانه	مقدار	توضیحات
۰	MAP		کنترل تغییر آدرس حافظه
۰			حالت راه‌انداز. قسمتی از ROM راه‌انداز به آدرس ۰ تغییر آدرس می‌دهد.
۱			حالت کاربر. حافظه فلش به آدرس ۰ تغییر آدرس می‌دهد.
۳۱:۱	-		از پیش تعیین‌شده. مقدار خوانده‌شده، تعریف‌شده است و تنها صفر باید نوشته شود.
			نامشخص

[۱] مقدار پس از راه‌اندازی مجدد