فهرست بخش‌های کتاب

عنوان **صفحه**

فصل

۵

آشنایی با نرم‌افزار کیل

نرم‌افزار کیل[۱] یک محیط برنامه‌نویسی و ویرایشگر برای میکروکنترلرها بوده و در سال ۱۹۸۲ توسط شرکت کیل تولید شده است. این نرم‌افزار در سال ۲۰۰۵ توسط شرکت ARM خریداری شد. شرکت ARM پس از خریداری این نرم‌افزار، تمرکز خود را در این نرم‌افزار بر روی میکروکنترلرهایی قرارداده که از هسته‌های این شرکت استفاده می‌کنند. این نرم‌افزار تمامی امکانات مورد نیاز برای برنامه نویسی میکروکنترلرهای ARM از جمله مترجم زبان‌های C و C++، خطایاب، شبیه‌ساز و محیط IDE را فراهم می‌کند. این نرم‌افزار از محیط برنامه‌نویسی و ویرایشگر uVision5 و مترجم armcc (ARM C/C++ Compiler) استفاده می‌کند. نرم‌افزار کیل از بیش از ۵۶۰۰ میکروکنترلر مبتنی بر هسته ARM که توسط بیش از ۳۲ شرکت مختلف تولید شده‌اند، پشتیبانی می‌کند.

برخی از قابلیت های نرم‌افزار کیل:

- پشتیبانی کامل از هسته‌های ARM7، ARM9، Cortex-M و غیره
- پشتیبانی از زبان های برنامه نویسی C، C++ و اسمبلی
- دارای محیط توسعه یکپارچه شامل محیط برنامه‌نویسی و ویرایشگر uVision5، شبیه‌ساز و خطایاب پیشرفته
- دارای مترجم پیشرو در صنعت با نام armcc
- دارای سیستم‌عامل بلادرنگ اختصاصی (RTX) به همراه کد منبع آن
- کلاس‌های برنامه‌نویسی مختلف برای کار با شبکه TCP/IP دستگاه‌های USB و غیره
- مجهز به کتابخانه‌های پیشرفته GUI برای کمک در نوشتن برنامه‌های گرافیکی
- پشتیبانی از خطایاب‌ها و برنامه‌ریزهای مختلفی همچون ST-Link، J-Link و غیره جهت برنامه‌ریزی میکروکنترلرها و خطایابی برنامه‌ها
- دارای مثال‌های آماده و گوناگون جهت آشنایی و یادگیری سریع‌تر

نرم‌افزار کیل دارای چهار نوع ویرایش است:

- MDK-Lite: نرم‌افزار کیل پس از نصب، به صورت پیش‌فرض دارای این ویرایش است. این ویرایش دارای محدودیت کد به اندازه 32KB بوده و برای بردها و تجهیزات آموزشی و پروژه‌های کوچک استفاده می‌شود.
- MDK-Essential: در این ویرایش فقط می‌توان برای هسته‌های سری Cortex-M برنامه نوشت و از برنامه‌نویسی امن در هسته‌های Cortex-M23 و Cortex-M33 حمایت نمی‌شود.
- MDK-Plus: این ویرایش علاوه بر ویژگی‌های ویرایش "MDK-Essential"، دارای کتابخانه‌هایی برای مواردی چون آدرس‌دهی IPv4، برنامه‌های گرافیکی، حالت میهمان USB و مدیریت فایل‌ها و پوشه‌ها است. در این ویرایش می‌توان برای هسته‌های سری ARM7، ARM9، هسته‌های امن، Cortex-M و Cortex-R4 برنامه نوشت و از برنامه‌نویسی امن در هسته‌های Cortex-M23 و Cortex-M33 نیز حمایت می‌شود.
- MDK-Professional: این ویرایش علاوه بر ویژگی‌های ویرایش "MDK- Plus"، از آدرس‌دهی IPv6، شبکه اینترنت اشیا و حالت میزبان USB و نیز از برنامه‌نویسی هسته‌هایی با معماری ARMv8-M حمایت می‌کند. نرم‌افزار کیل در این ویرایش دارای تمامی امکانات ارائه شده توسط شرکت ARM است.

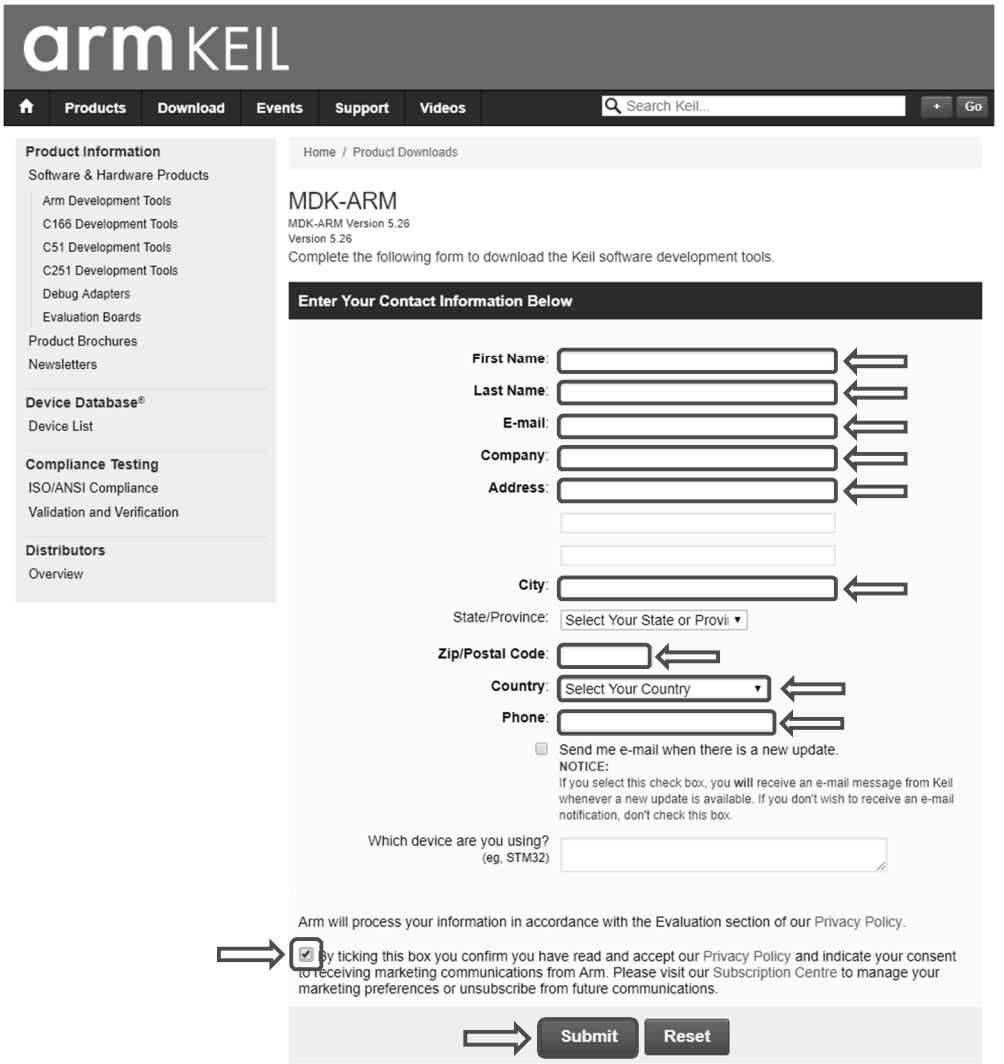
۵/۱ چگونگی دریافت و نصب نرم‌افزار کیل و بسته‌های آن

۵/۱/۱ چگونگی دریافت نرم‌افزار کیل

نگارش ۵/۲۶ این نرم‌افزار در DVD کتاب و در آدرس "Compiler/Keil" و با نام "MDK526.EXE" قراردادشده است. در صورتی که می‌خواهید آخرین نگارش آن را دریافت کنید، ابتدا به آدرس "**https://www.keil.com/demo/eval/arm.htm**" بروید. در این آدرس با صفحه‌ای مانند شکل بعد روبرو می‌شوید، که می‌بایست موارد مشخص شده را پرکنید.

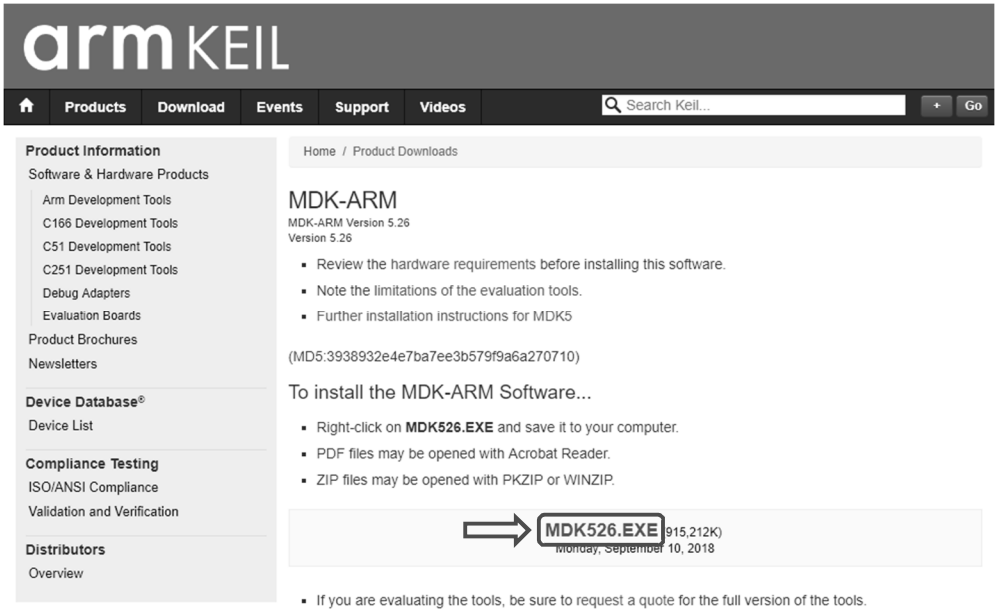
---

۱. تلفظ انگلیسی این کلمه به صورت "کایل" بوده و "کیل" فارسی شده آن می‌باشد.



شکل ۱. صفحه ورود مشخصات برای دریافت فایل نرم‌افزار کیل

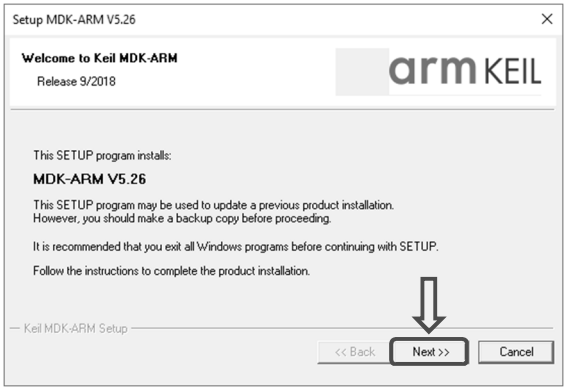
با زدن برروی کلید "Submit" به صفحه بعد می‌رسیم. در این صفحه با زدن بر روی نام فایل و دادن آدرس دلخواه برای ذخیره شدن بر روی رایانه تان، دریافت نرم‌افزار کیل آغاز می‌شود.



شکل ۲. صفحه دریافت فایل نرم‌افزار کیل

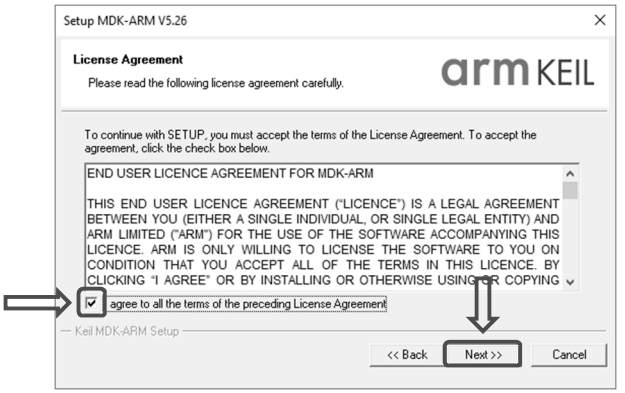
۵/۱/۲ چگونگی نصب نرم‌افزار کیل

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



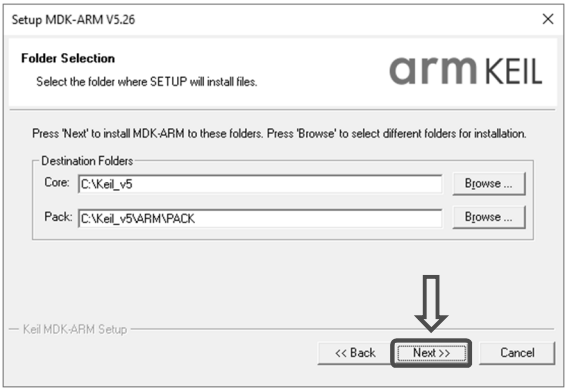
شکل ۳. پنجره نصب نرم‌افزار کیل

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



شکل ۴. پنجره قوانین و شرایط استفاده

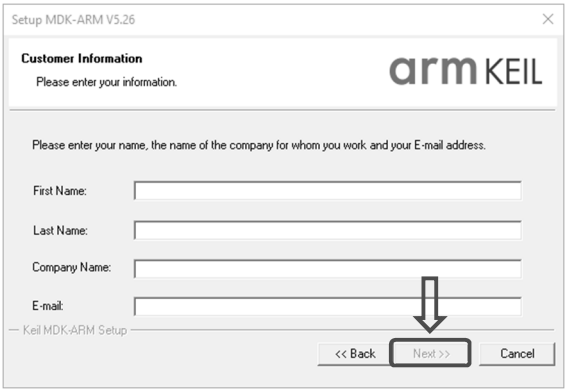
با انتخاب جعبه بررسی، قوانین و شرایط استفاده را پذیرفته و سپس بر روی کلید Next می‌زنیم تا پنجره محل نصب نرم‌افزار کیل ظاهر شود.



شکل ۵. پنجره محل نصب نرم‌افزار کیل

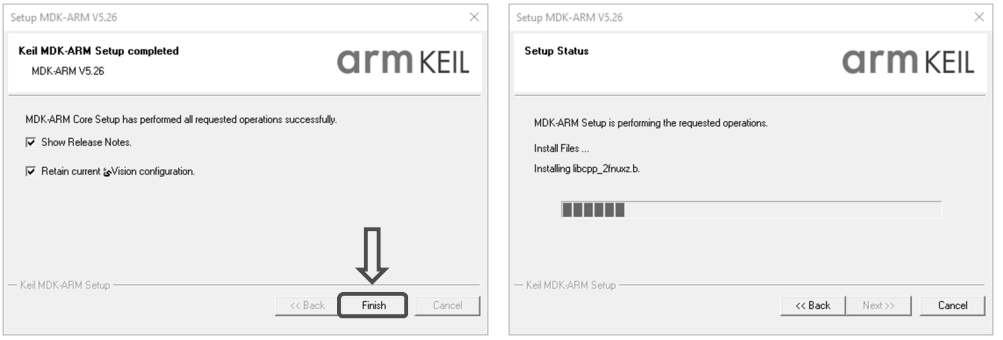
تذکر: شما می‌توانید آدرس محل نصب نرم‌افزار کیل را تغییر دهید اما توصیه می‌کنیم آنرا تغییر ندهید زیرا بسیاری از برنامه‌های نوشته شده توسط دیگران از همین آدرس پیش‌فرض برای دسترسی به بعضی از فایل‌های مورد نیازشان استفاده می‌کنند و در صورت تغییر این آدرس، دیگر نمی‌توانند به آن فایل‌ها دسترسی داشته باشند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



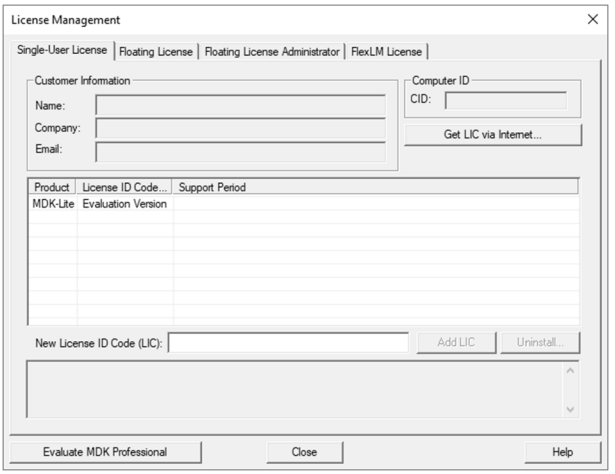
شکل ۷. پنجره شروع فرآیند نصب (سمت راست) و پنجره پایان فرآیند نصب (سمت چپ)

با پایان فرآیند نصب بر روی کلید Finish می‌زنیم.

۵/۱/۳ چگونگی کرک نرم‌افزار کیل

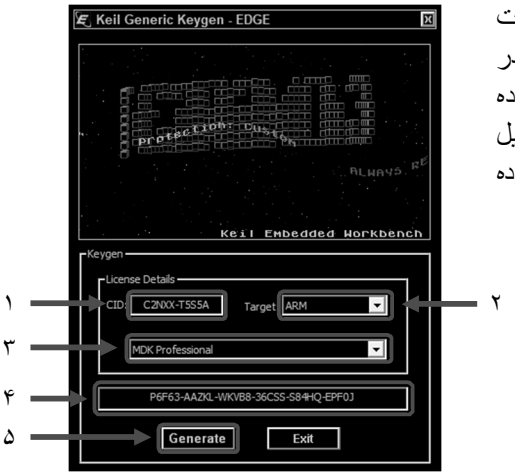
نرم‌افزار کیل به صورت پیش فرض دارای محدودیت ترجمه برنامه تا حجم 32KB می‌باشد که برای برداشتن این محدودیت می‌بایست به آن گواهینامه دهیم. برای این کار ابتدا نرم‌افزار کیل را طبق بخش ۵/۱/۴ اجرا می‌کنیم. سپس

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



شکل ۸. پنجره مدیریت گواهینامه‌ها

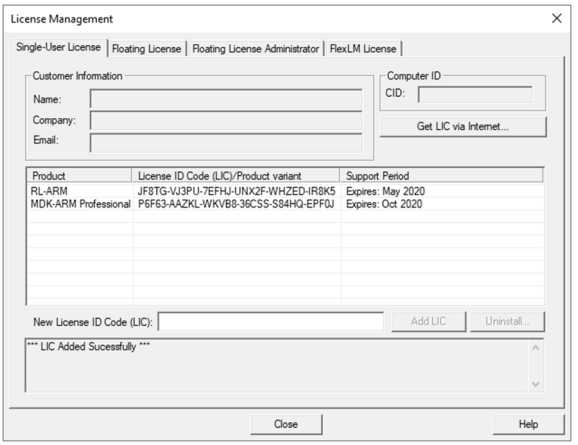
در برگه "Single-User License" از قسمت "Computer ID" بخش CID را کپی می‌کنیم. سپس در DVD کتاب و در آدرس "Compiler/Keil" فایل فشرده "KeyGen v5.xx.7z" را اجرا می‌کنیم. در این فایل، دو فایل ایجادکننده گواهینامه وجود دارد که یکی از آنها را اجرا کرده و درنتیجه پنجره‌ای همانند شکل روبرو باز می‌شود.



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 Real-Time OS" را انتخاب و بقیه مراحل را مثل قبل تکرار می‌کنیم. بدین ترتیب، پنجره مدیریت گواهینامه‌ها باید به صورت روبرو درآمده باشد.

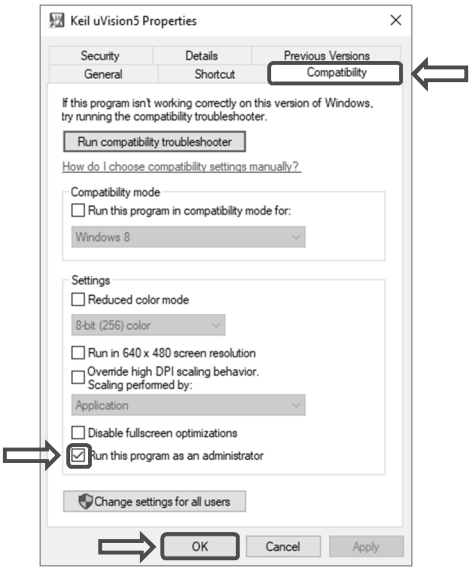


شکل ۱۰. پنجره مدیریت گواهینامه‌ها پس از دادن گواهینامه به نرم‌افزار کیل

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

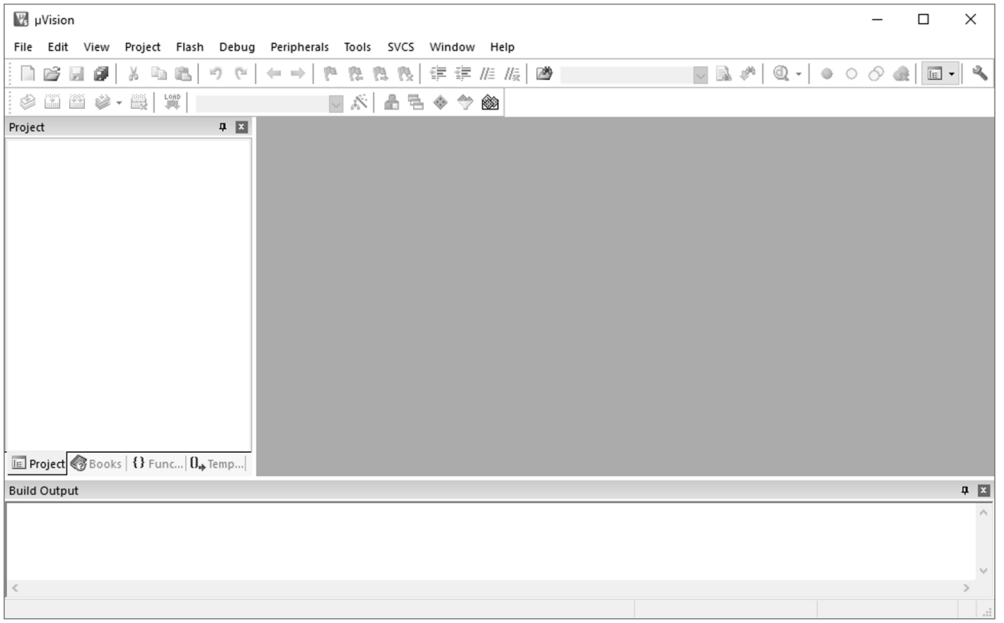
اگر برای اولین بار می‌خواهید نرم‌افزار کیل را اجرا کنید ابتدا بر روی میانبر نرم‌افزار (ایجاد شده پس از نصب بر روی دسکتاپ با نام "Keil uVision5") راست کلیک کرده و از پنجره باز شده و در انتهای آن، گزینه Properties را انتخاب کنید. در این هنگام پنجره‌ای همانند شکل روبرو باز می‌شود.



شکل ۱۱. پنجره Properties

در پنجره بازشده 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.

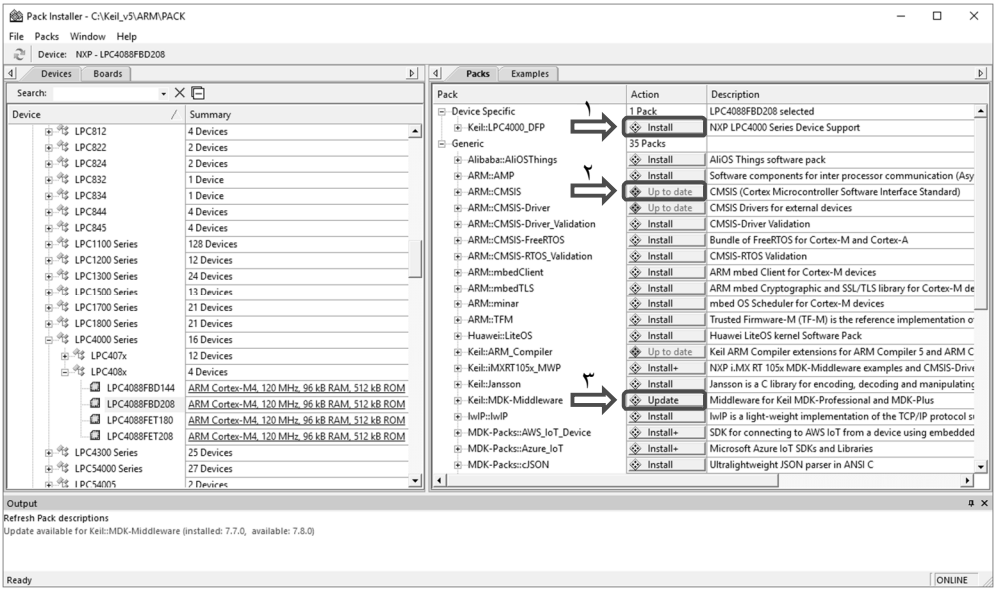


شکل ۱۲. نرم‌افزار کیل

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:

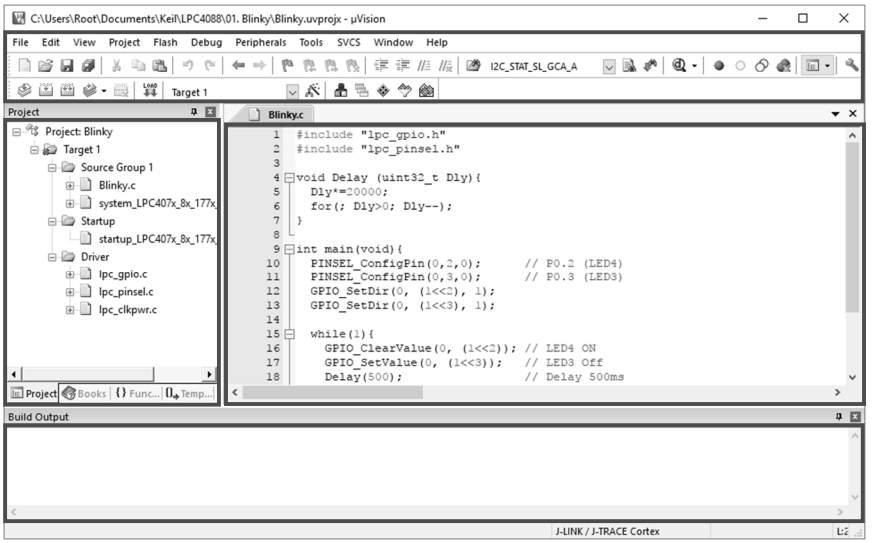
- **گزینه (۱) دریافت و نصب:** بسته موردنظر دریافت و نصب می‌شود.
- **گزینه (۲) بروزبودن:** نشان دهنده آن است که برای آن مورد، جدیدترین بسته نرم‌افزاری نصب شده است.
- **گزینه (۳) بروزرسانی:** نشان دهنده آن است که بسته نرم‌افزاری نصب شده برای آن مورد، قدیمی بوده اما نگارش بروزتری برای آن وجود دارد و می‌تواند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

۵/۲/۲ توضیحاتی در مورد منوی نرم‌افزار کیل

منوی نرم‌افزار کیل 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 کیل
- View: شامل گزینه‌هایی برای نمایش/عدم نمایش هر کدام از پنجره‌ها، نوارابزارها و نوار وضعیت در کیل
- 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 پیکره‌بندی و اجرای نرم‌افزارهای جانبی (مربوط به دیگر شرکت‌ها) در کیل
- SVCS: پیکره‌بندی ویرایشگر کیل برای استفاده از نرم‌افزار کنترل نگارش
- Window: شامل گزینه‌هایی برای مدیریت بخش ویرایشگر
- Help: شامل گزینه‌هایی مانند راهنمای نرم‌افزار، بررسی بروزرسانی و غیره

۵/۲/۳ توضیحاتی در مورد نوارابزارهای نرم‌افزار کیل

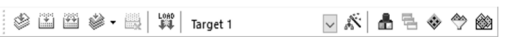
نرم‌افزار کیل دارای دو نوارابزار File و Build است.

نوارابزار 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 ویا 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 ویا 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.

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



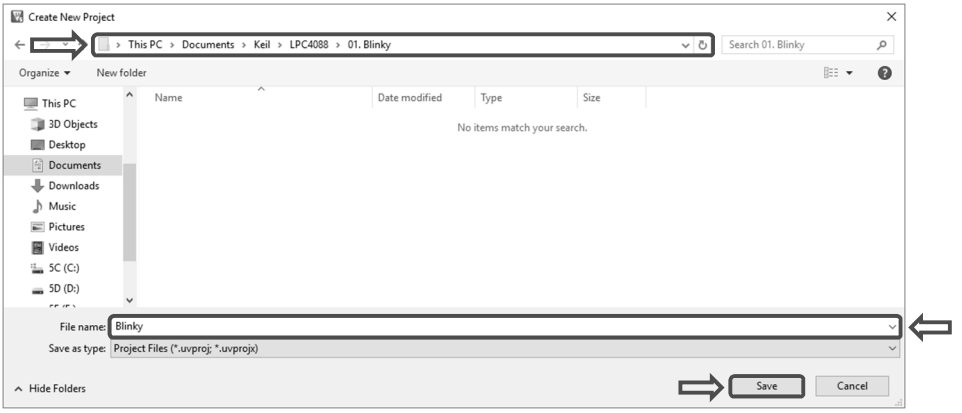
- Translate: از این ابزار برای ترجمه فایل درحال نمایش در بخش ویرایشگر، استفاده می‌شود. کلید میانبر آن "Ctrl+F7" است.
  تذکر: این ابزار فقط یک فایل را ترجمه می‌کند.
- Build: این ابزار فقط فایل‌های تغییریافته را ترجمه کرده و درنهایت فایل hex برنامه را ایجاد می‌کند. کلید میانبر آن "F7" است.
- Rebuild: این ابزار همه فایل‌های برنامه را ترجمه کرده و درنهایت 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 تغییریافته را ترجمه کرده و درنهایت فایل hex همه برنامه‌ها را ایجاد می‌کند. دومین گزینه (Batch Rebuild) تمامی برنامه‌های انتخاب شده را به صورت کامل ترجمه کرده و درنهایت فایل hex همه برنامه‌ها را ایجاد می‌کند. سومین گزینه (Batch Clean) فایل‌های اضافی تولیدشده در هنگام ترجمه برنامه توسط کیل در دو 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 نرم‌افزار کیل استفاده می‌شود.

۵/۳ ایجاد پروژه و تنظیمات آن در نرم‌افزار کیل

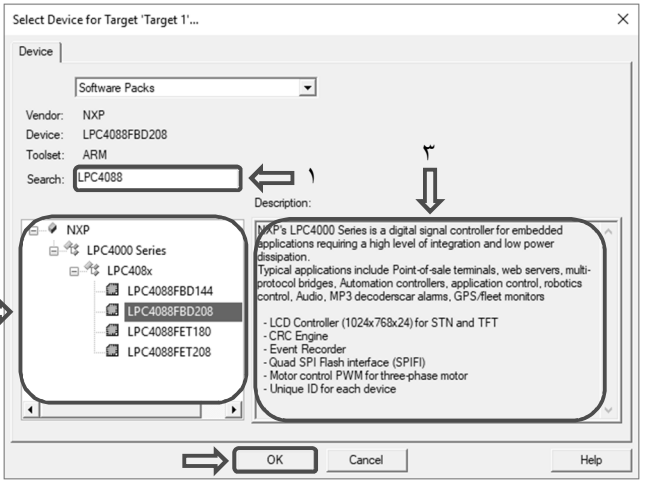
۵/۳/۱ چگونگی ایجاد پروژه

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"

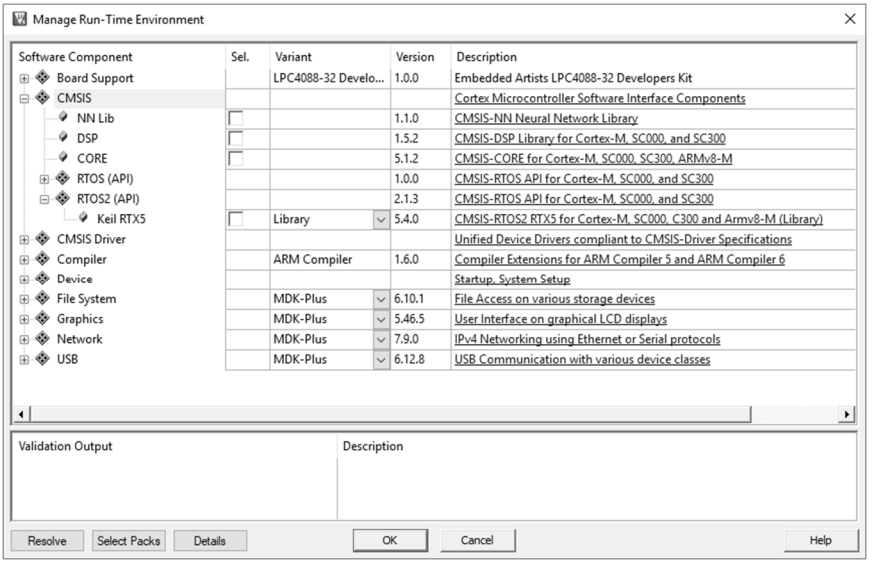
در این پنجره مسیر پروژه را انتخاب کرده، برای پروژه نامی در نظر گرفته و سپس آنرا ذخیره می‌کنیم. پس از ذخیره شدن پروژه، پنجره شکل بعد به صورت خودکار باز می‌شود.



شکل ۱۶. پنجره انتخاب تراشه

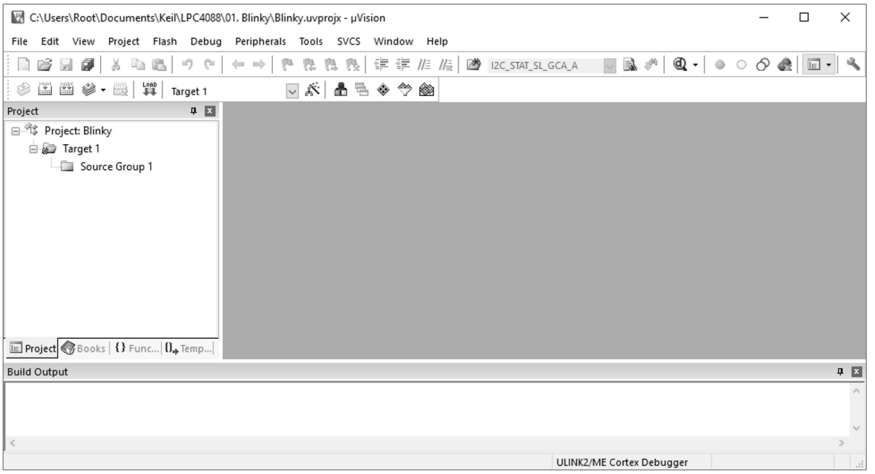
پنجره شکل بالا، پنجره انتخاب تراشه بوده و بنابراین برای انتخاب تراشه یکی از دو کار زیر را می‌توان انجام داد ۱. جستجوی نام کامل تراشه و سپس انتخاب آن در قاب (۲)، ۲. نوشتن بخشی از نام تراشه در قاب (۱) و انتخاب نام کامل تراشه در قاب (۲). روش دوم ساده‌تر بوده و در هر دو حالت پس از انتخاب نام کامل تراشه،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



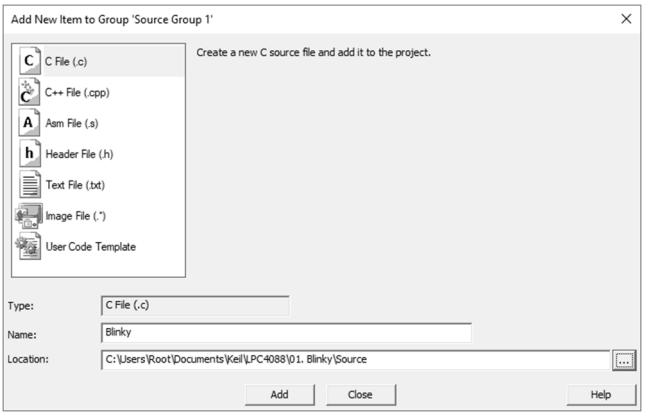
شکل ۱۷. پنجره انتخاب امکانات مورد نیاز

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 Item to Group 'Source Group 1'" را انتخاب می‌کنیم. درنتیجه 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
  }
}
```

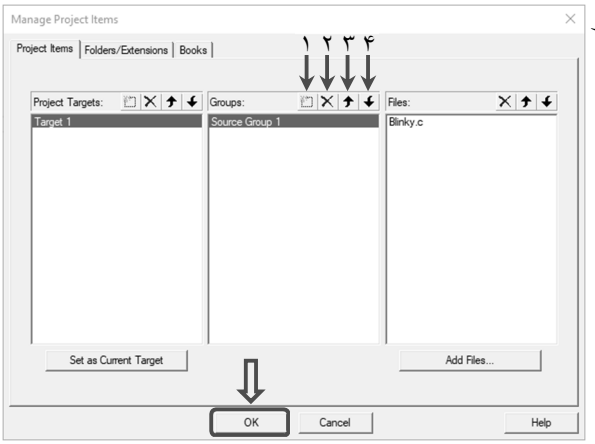
پس از نوشتن برنامه، از منوی File گزینه Save (Ctrl+S) را انتخاب کرده و برنامه را ذخیره می‌کنیم. در ادامه برای کارکرد صحیح برنامه باید چند فایل به نمودار درختی برنامه اضافه شود. این فایل‌ها عبارتند از:

- 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 ذکرشده 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 ذکرشده در زیر، به برنامه خود اضافه کنید.

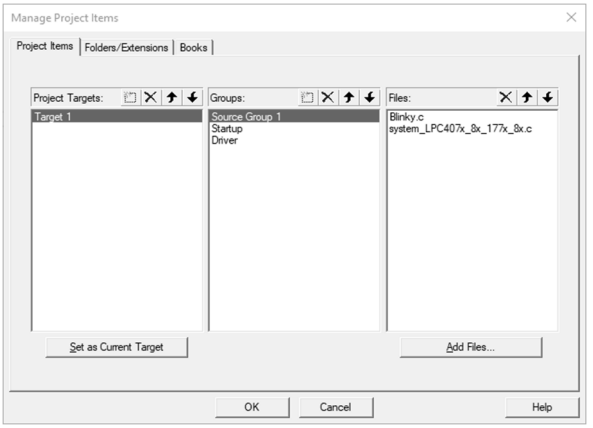
برای اضافه کردن این پنج فایل ابتدا از منوی Project، گزینه Manage و سپس گزینه "Project Items" را انتخاب

کنید. درنتیجه 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) 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.



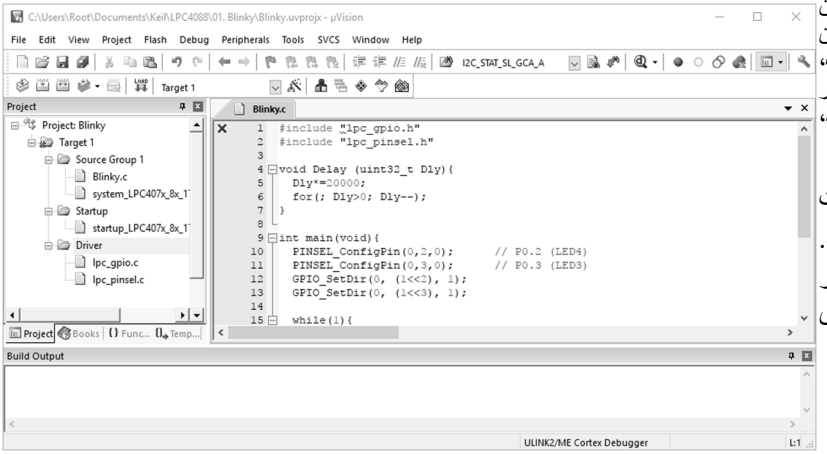
شکل ۲۱. پنجره مدیریت پروژه

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.

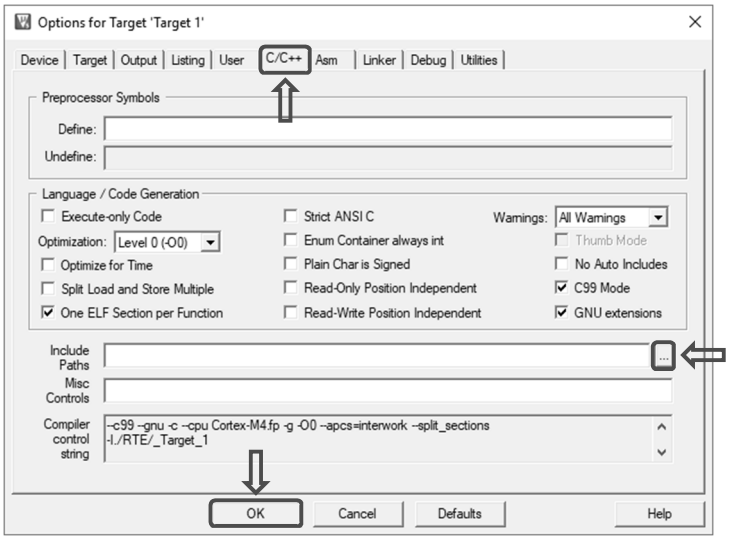
تذکر ۲: همه فایل‌ها را می‌توان در یک پوشه قرار داد، اما این کار در پروژه‌های بزرگ، مدیریت پروژه را سخت‌تر می‌کند. با پوشه‌بندی پروژه، مدیریت آن آسان‌تر می‌شود.

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



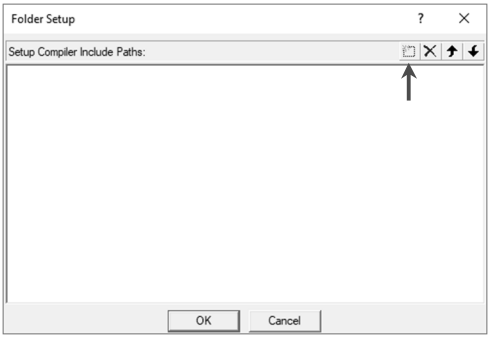
شکل ۲۲. نمایی از پروژه در نرم‌افزار کیل

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 درنتیجه پنجره "Options for Target" باز می‌شود. در این پنجره همچون شکل زیر، سربرگ "C/C++" را انتخاب می‌کنیم.



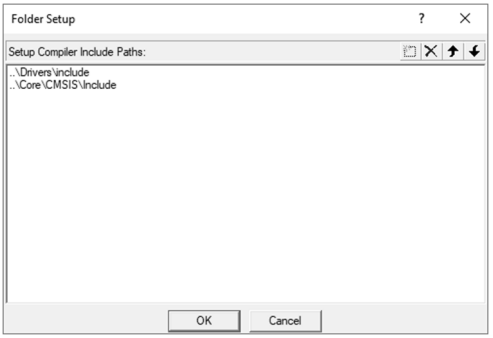
شکل ۲۳. سربرگ C/C++ در پنجره "Options for Target"

در این سربرگ کلید "..." را می‌زنیم. درنتیجه 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. درنهایت 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. درنهایت باید آن علامت ضربدر قرمز

رنگ حذف شده باشد. نوشتن برنامه در این مرحله کامل شده و برنامه آماده ترجمه می‌باشد.

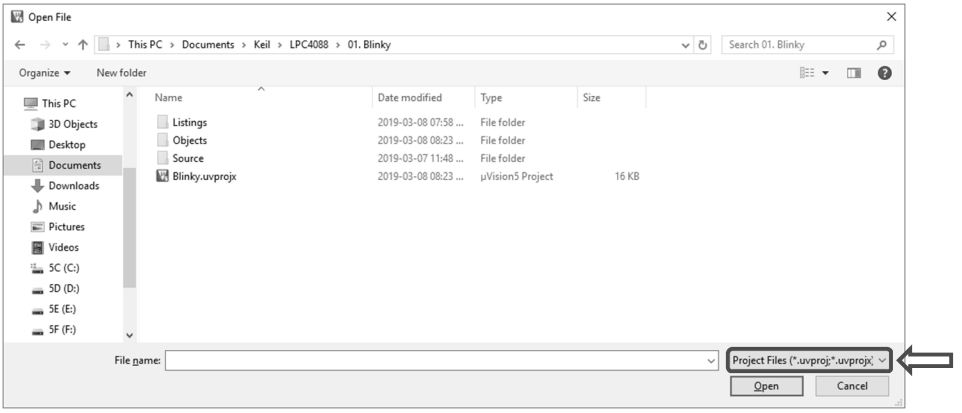
تذکر: برای کارکرد صحیح برنامه پس از انتقال به برد مدار چاپی باید در پنجره "Options for Target" و در سربرگ Target، مقدار نوسانگر وصل شده به تراشه را در قاب "Xtal (MHz)" تنظیم کنیم. در این سربرگ تنظیمات حافظه متناسب با نیاز برنامه باید انجام شود که در برنامه چشمک‌زن، بدلیل سادگی، نیازی به تنظیمات خاصی 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[۲]
- هسته‌های امن SC000 و SC300

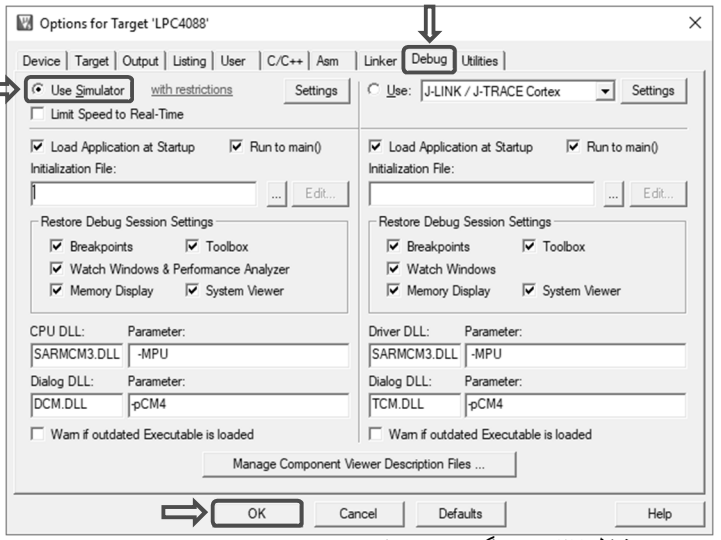
تذکر: برای شبیه‌سازی هسته‌های قدیمی مانند ARM7 و ARM9 می‌بایست فایل "MDK79525.EXE" در DVD کتاب و از آدرس "Compiler\Keil\Software Pack" نصب شود.

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

۲. Fixed Virtual Platform

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

برای استفاده از حالت شبیه‌سازی در نرم‌افزار کیل، 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"

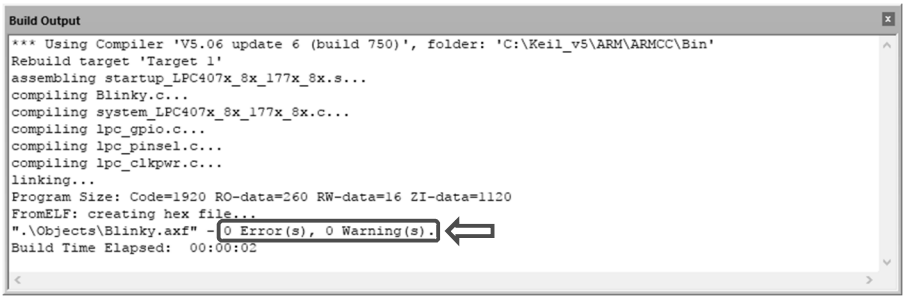
تذکر: در پنجره شکل بالا، سمت راست پنجره مربوط به خطایابی از طریق سخت‌افزار بوده و سمت چپ پنجره مربوط به خطایابی از طریق نرم‌افزار (خود نرم‌افزار کیل) می‌باشد.

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 نوارابزار 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.



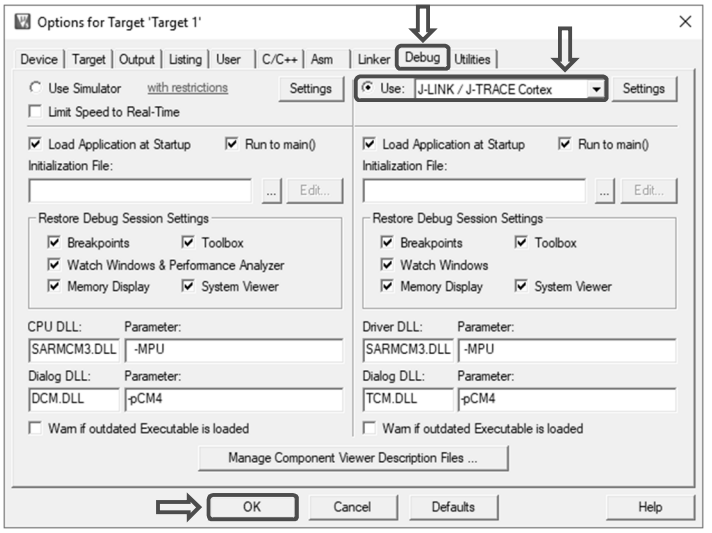
شکل ۲۸. پنجره "Build Output"

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.

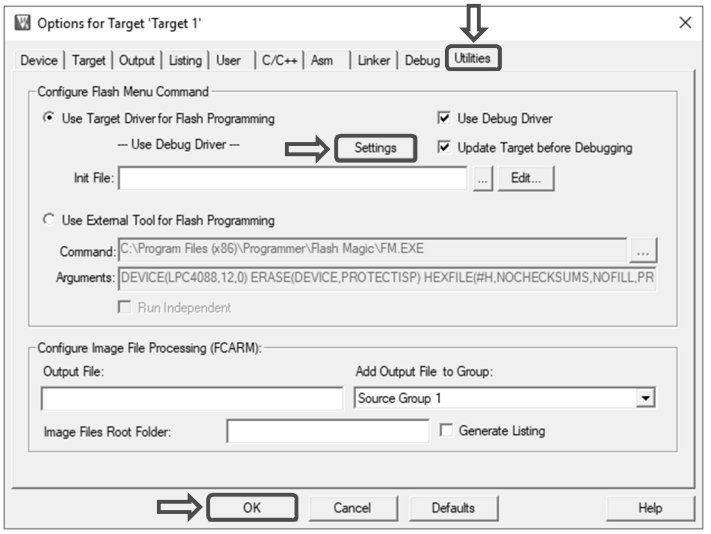
۵/۴/۲ انتقال پروژه به برد مدار چاپی

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.



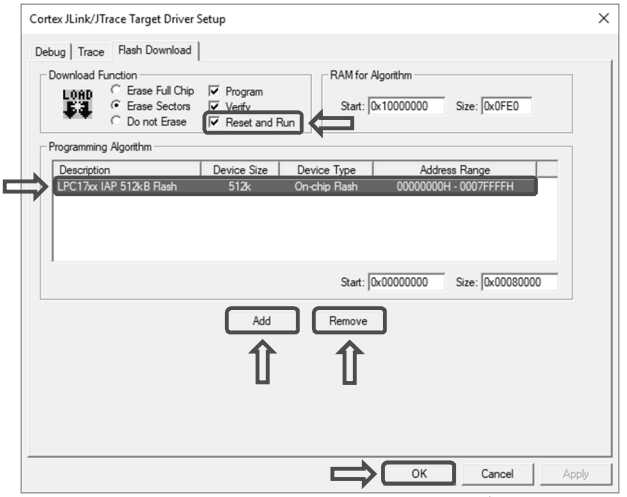
شکل ۲۹. سربرگ Debug در پنجره "Options for Target"

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



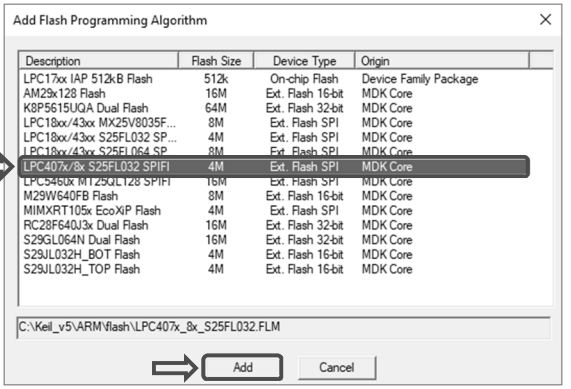
شکل ۳۰. سربرگ Utilities در پنجره "Options for Target"

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



شکل ۳۱. پنجره انتخاب تراشه برای برنامه‌ریز

در پنجره شکل قبل در قسمت "Download Function"، جعبه 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. درنتیجه 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 برنامه‌ریز، کلید OK را زده و بعد از آن در سربرگ Utilities نیز کلید OK را می‌زنیم.

پس از تنظیم برنامه و هنگامی‌که برنامه‌ریز به رایانه وصل است، از منوی Flash گزینه Download (F8) را انتخاب می‌کنیم. سپس نرم‌افزار کیل از طریق برنامه‌ریز با برد مدار چاپی ارتباط برقرار کرده و فایل hex را به تراشه روی برد منتقل می‌کند.

فصل

۲۲

خطایابی و ردیابی

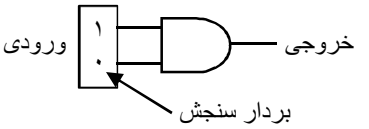
واژه‌های کلیدی:

JTAG	درگاه دسترسی به
انجمن مهندسان برق و الکترونیک	

TAP: Test Access Port
IEEE: Institute of Electrical and Electronics Engineers

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 پشت‌سرهم 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 $Q = ۲^۲ = ۴$ بردار سنجش است. در مداری با ۲۵ ورودی و ۵۰ قطعه/بخش حافظه دار، Q برابر با $۳/۸\times۱۰^{۲۲} \approx ۲^{۷۵}$ حالت خواهد شد و اگر پردازنده با سرعت 1GHz کار کند، برای بررسی همه این حالت‌ها تقریبا 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 ویا 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

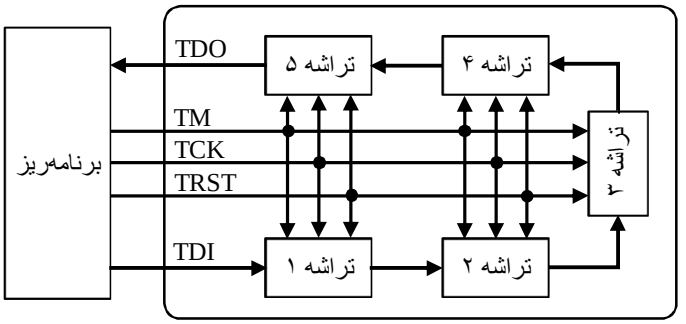
- حمایت از حالت‌های JTAG استاندار و خطایابی سیمی سریالی (SWD)
- دسترسی مستقیم از طریق خطایابی به تمامی حافظه‌ها، ثبات‌ها و اجزای جانبی
- لازم نداشتن تراشه/دستگاه به هیچ تجهیزات اضافی برای خطایابی
- قابلیت ردیابی دستورهای در حال اجرا در پردازنده توسط درگاه ردیابی (Trace)
- دارای هشت نقطه توقف. شش نقطه 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 بیدارنشدن تراشه از حالت‌های خواب عمیق و کاهش مصرف توان به روش معمول بخاطر ترکیب‌شدن بخش خطایابی با هسته Cortex-M4، درنظر 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 معمولا 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 برنامه کاربر استفاده شوند.

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

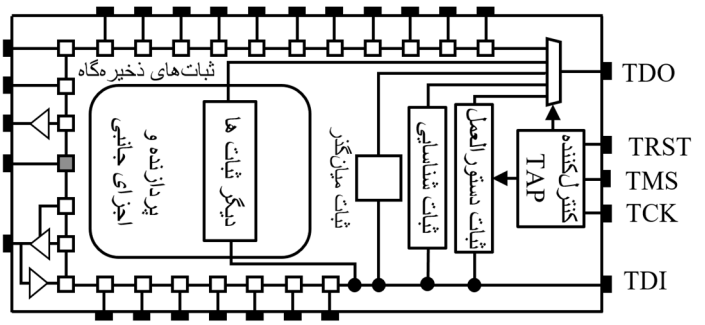
نام پایه	نوع	توضیحات
پایه‌های حالت JTAG		
JTAG_TCK	ورودی	**ضربان JTAG.** این پایه زمانی که تراشه در حالت JTAG قرار دارد، تامین‌کننده ضربان بخش خطایابی است. ضربان دریافتی از این پایه باید ۶ برابر کمتر از ضربان پردازنده (CCLK) باشد.

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	**راه‌اندازی‌مجدد JTAG.** از این پایه می‌توان برای راه‌اندازی‌مجدد بخش خطایابی/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، تامین‌کننده ضربان بخش خطایابی 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، تامین‌کننده ضربان بخش ردیابی است.
TRACEDATA[3:0]	خروجی	**داده ردیابی (بیت‌های ۰ تا ۳).** داده حالت ردیابی ETM به صورت فشرده‌شده از طریق این پایه‌ها به برنامه‌ریز ارسال می‌شود.

[۱] Programmer

۲۲/۵ خطایابی 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 خطایابی JTAG بوده و شامل موارد زیر است:

- مجموعه‌ای از چهار پایه اختصاصی (TDI، TMS، TCK و TDO) و پایه انتخابی[۱] TRST
- یک ثبات ذخیره‌گاه، یک ثبات دستور و چند ثبات داده (شامل ثبات میان‌گذر ۱-بیتی و ثبات شناسایی ۳۲-بیتی)
- یک کنترل‌کننده TAP

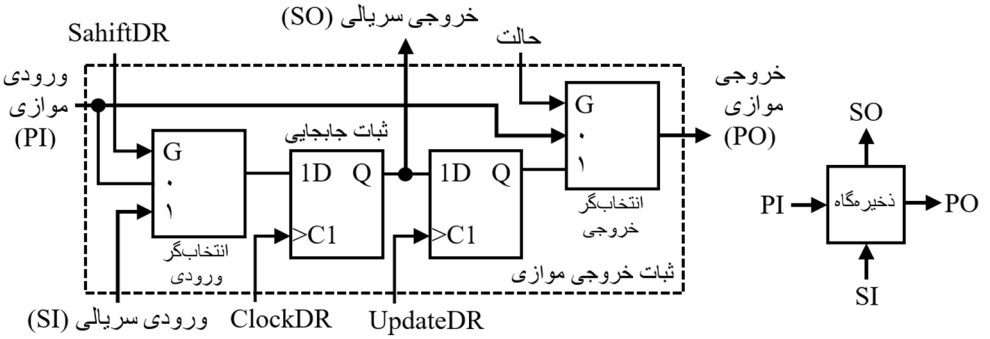
۲۲/۵/۱ ذخیره‌گاه

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 دوجهته و پایه‌های سه حالته 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 ذخیرگاه قبلی و ذخیرگاه بعدی (در زنجیره) ارتباط برقرار کند.
ذخیرگاه‌ها با هم در ارتباط بوده و تشکیل زنجیره‌ای از ذخیرگاه‌ها 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 ذخیرگاه" 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 ذخیرگاه بوده ولی اگر در کنار قابلیت آنالوگ دارای قابلیت دیجیتال نیز باشد، به دلیل قابلیت دیجیتال دارای ذخیرگاه نیز خواهد بود. هر پایه ورودی/خروجی حداقل دارای دو ذخیرگاه (یکی برای قسمت ورودی و یکی برای قسمت خروجی) است. ذخیرگاه قسمت ورودی، داده را از پایه تراشه و از طریق PI دریافت و آنرا از طریق PO به پردازنده ارسال می‌کند. ذخیرگاه قسمت خروجی، داده را از پردازنده و از طریق PI دریافت و آنرا از طریق PO به پایه تراشه ارسال می‌کند. هر ذخیرگاه 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) ذخیرگاه بعدی منتقل و از طریق SI خود، داده را از SO ذخیرگاه قبلی دریافت کند.



شکل ۴. ذخیرگاه 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 باارزش 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

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

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.

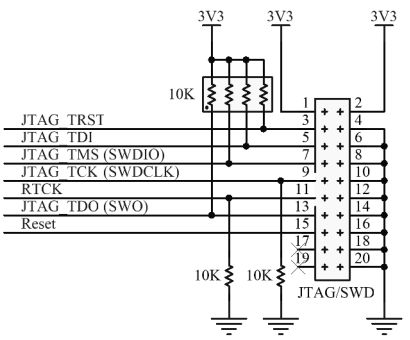
ثبات‌های داده: ثبات‌های داده طبق استاندارد JTAG شامل ثبات‌های میان‌گذر و شناسایی بوده و در پردازنده‌های ARM، شامل ثبات‌های میان‌گذر، شناسایی، عدم موفقیت، DPACC و APACC می‌باشند. ثبات‌های داده عبارتند از:

- **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 متوقف‌کردن 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.

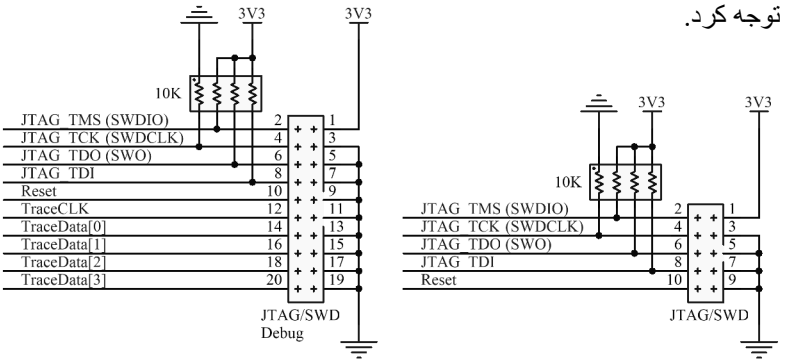
۲۲/۵/۳ کانکتورهای مورد استفاده برای خطایابی و ردیابی

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 به‌صورت بالاکش در درون تراشه بکار رفته برای آن درنظرگرفته‌شده‌اند.



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_swdclk    : 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

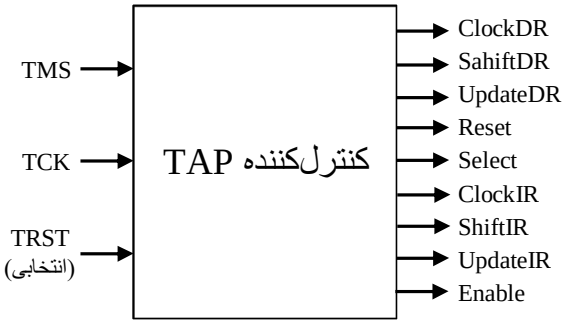
```
attribute IDCODE_REGISTER of LPC408x : entity is
"0xxx"              & -- Version Number
"1111111111000000"  & -- Part Number
"00000010101"       & -- Manufacturer ID
"1                  -- Required by IEEE
```

- 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 (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 راه‌اندازی‌مجدد بخش خطایابی استفاده می‌شود.
کنترل‌کننده TAP مسئول کنترل کامل بخش خطایابی در تراشه است. کنترل‌کننده، فعال یا غیرفعال‌شدن 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 میان‌گذر، 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 راه‌اندازی‌مجدد 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 خطایاب، اجرای پردازنده را بلافاصله بعد از راه‌اندازی‌مجدد متوقف می‌کند اما ROM راه‌انداز هنوز شروع آدرس آن از ۰ است و می‌تواند باعث دردسر شود. در حالت ایدآل، خطایاب 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	م.پ.ر[۱]
۰	MAP		کنترل تغییر آدرس حافظه	۰
		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 خواندشده، تعریف‌نشده است و تنها صفر باید نوشته شود.	نامشخص

[۱] مقدار پس از راه‌اندازی‌مجدد