



دانشگاه زنجان

دانشگاه مهندسی

گروه برق

پایان نامه کارشناسی

گرایش: الکترونیک

عنوان: خود آموز استودیو کدنگار TMS320C6000

استاد راهنما: دکتر اصغر طاهری

تاریخ دفاعیه: خرداد ۹۱

۱ نگاه کلی به استودیو کدننگار

- ۱.۱ توسعه استودیو کدننگار
- ۱.۲ ابزارهای تولید کد
- ۱.۳ استودیو کدننگار به همراه IDE
 - ۱.۳.۱ ویژگی تصحیح کد
 - ۱.۳.۲ ویژگی تولید و ساخت برنامه کاربردی
 - ۱.۳.۳ ویژگی عیب یابی
- ۱.۴ اتصالات DSP/Bios
- ۱.۵ نمونه سازی سخت افزاری و تبادل اطلاعات بلادرنگ
- ۱.۶ اتصالات ثالثیه
- ۱.۷ متغیرها و فایل های کدننگار
 - ۱.۷.۱ پوشه های نصب
 - ۱.۷.۲ پسوند فایل ها
 - ۱.۷.۳ متغیرهای محیطی
 - ۱.۷.۴ توسعه محیط DSP

۲ ساخت یک برنامه ساده

- ۲.۱ تولید یک پروژه جدید
- ۲.۲ بازیابی کد
- ۲.۳ مرور کد
- ۲.۴ ساخت و اجرای برنامه
- ۲.۵ تغییرات گزینه های برنامه و تصحیح خطاهای دستوری
- ۲.۶ استفاده از نقطه شکست ها و پنجره های نظارتی
- ۲.۷ استفاده از پنجره نظارتی با ساختارها
- ۲.۸ زمان بندی اجرای کد برنامه
- ۲.۹ مثال های اضافه

۳ ایجاد یک برنامه DSP/Bios

دو

۵.۹ مثال های اضافی

۶ آنالیز رفتار های بلا درنگ

۶.۱ گشایش و امتحان پروژه

۶.۲ اصلاح فایل پیکربندی

۶.۳ مشاهده تغییرات سورس کد

۶.۴ استفاده از کنترل RTDX برای تغییر بار در زمان اجرا

۶.۵ تصحیح اولویت وقفه سخت افزاری

۶.۶ مثال های بیشتر

۷ اتصال ابزار های I/O

۷.۱ گشایش و آزمایش پروژه

۷.۲ مشاهده سورس کد C

۷.۳ مشاهده برنامه کاربردی Signalprog.exe

۷.۴ اجرای برنامه

۷.۵ اصلاح سورس کد برای استفاده از کانال های میزبان و لوله ها

۷.۶ چندی در مورد کانال های میزبان و لوله ها

۷.۷ اضافه نمودن کانال ها و SWI به فایل پیکربندی

۷.۸ اجرای برنامه اصلاح شده

۷.۹ مثال های بیشتر

مقدمه:

استودیو کدننگار توسعه فرآیند برنامه‌نویسی برای برنامه‌های کاربردی پردازش سیگنال‌های جاسازی شده و بلادرنگ را تولید و تسهیل می‌نماید. استودیو کدننگار توانایی‌های کدننگار IDE که شامل آگاهی کامل برد هدف از DSP توسط میزبان و گسترش آنالیزهای بلادرنگ است، می‌باشد.

این خود آموز با فرض اینکه شما، کدننگار شامل ابزارهای تولید کد TMS320C6000 به همراه APIها و اتصالات برای DSP/Bios و RTDX می‌باشد دارا هستید، تهیه و تولید شده است. این خودآموز همچنین فرض می‌کند که شما برد مرجع را که شامل ابزار DSP است در رایانه خود نصب نموده اید.

اگر از شبیه ساز DSP به جای برد استفاده می‌کنید شما محدود به گام‌های موجود در قسمت دوم و چهارم هستید.

این خودآموز برخی از نکات کلیدی را معرفی می‌کند. و هدف یک توضیح کلی و فراگیر از هر ویژگی نمی‌باشد. و منظور آن آماده‌سازی شما برای توسعه DSP در استودیو کدننگار می‌باشد.

قراردادهای نوشتاری:

هسته‌ی TMS320C6000 به عنوان C6000 خوانده می‌شود.

کدننگار فایل‌هایی با پسوند s62 و h62. تولید می‌کند که می‌تواند در TMS320c6201 و TMS320C6701 استفاده شوند. DSP/Bios از دستورات شناور که توسط TMS320C6701 پشتیبانی می‌شود، استفاده نمی‌کند.

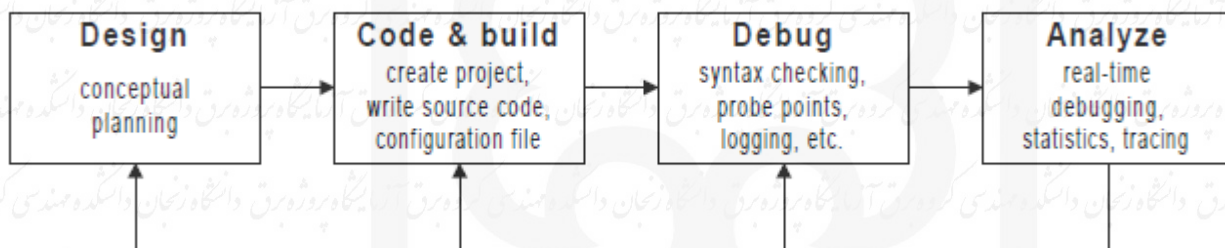
براکت‌ها یک پارامتر نوشتاری را مشخص می‌کنند. اگر شما یک پارامتر اختیاری را انتخاب کنید، می‌توانید اطلاعات را داخل براکت مشخص نمایید. تا وقتی که براکت‌ها بولد نشده‌اند اطلاعات وارد ننمایید.

نگاه کلی به استودیو کدنگار

این قسمت یک نمای کلی از فرآیند توسعه نرم افزاری، اجرای کدنگار، فایل ها و متغیرهای استفاده شده در این قسمت می باشد. استودیو کدنگار فرآیند توسعه برای برنامه نویسانی که کار تولید و آزمایش برنامه های پردازش سیگنال بلادرنگ و جاسازی شده را بر عهده دارند، سرعت و تسهیل می بخشد. این برنامه ابزارهایی برای پیکربندی، ساخت، عیب یابی، دنباله روی و آنالیز برنامه ها را تامین می کند.

۱.۱ توسعه استودیو کدنگار

استودیو کدنگار ابزار های پایه ی تولید کد با یک سری از توانایی های آنالیز بلادرنگ و عیب یابی را فراهم می کند. استودیو کدنگار تمام فازهای سیکل را پشتیبانی می کند.



به منظور استفاده از این خودآموز بایستی کارهای زیر را انجام دهید:

☐ **نصب برد هدف و درایور نرم افزار:** نکات نصب به همراه برد را دنبال کنید و اگر از یک شبیه ساز یا بردی که درایور آن توسط این محصول تامین شده است استفاده می کنید، می توانید به گام های راهنمایی نصب این محصول مراجعه نمایید.

☐ **نصب استودیو کدنگار:** راهنمایی های نصب را دنبال کنید و اگر از شبیه ساز کدنگار و ابزارهای تولید کد TMS320C6000 استفاده می کنید از گام دوم در بخش ۲ و ۴ اقدام نمایید.

☐ **نصب و اجرای کدنگار:** برنامه ی نصب شده اجازه ی استفاده از درایور نصب شده روی برد مرجع را می دهد.

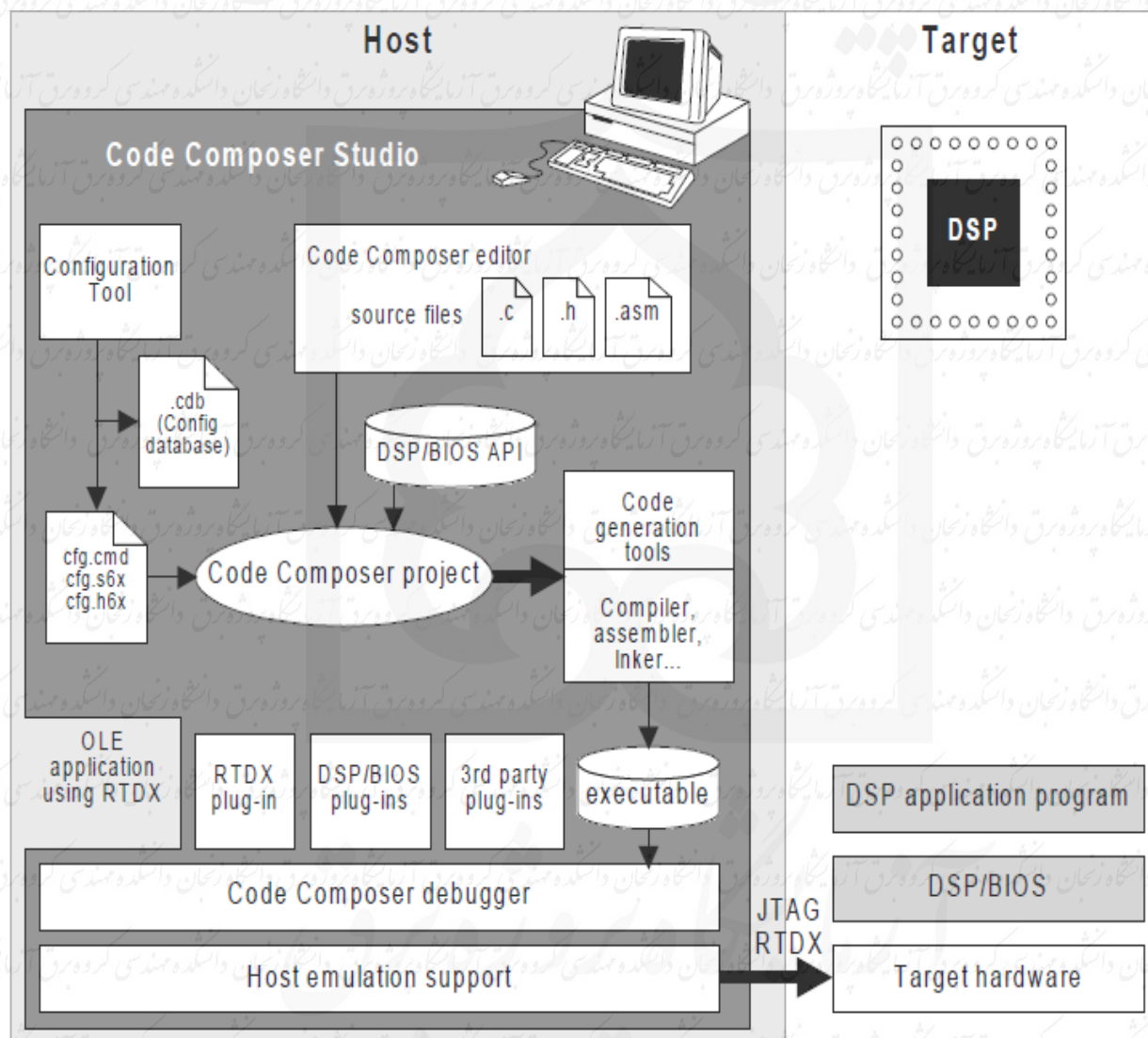
کدنگار شامل اجزای زیر می باشد:

□ اجزای تولید کد TMS320C6000.

□ IDE کدنگار اتصالات DSP/Bios و API

□ اتصالات RTDX، رابط میزبان و API

اتصالات همانند شکل زیر کار می کنند:



۱-۲ ابزارهای تولید کد:

ابزارهای تولید کد، توسعه کدنगर را فراهم می آورد. شکل صفحه بعد یک جریان پیشرفت نمونه را نشان می دهد. مهم ترین راه های توسعه نرم افزاری برای برنامه های به زبان C سایه خورده اند. قسمت های دیگر انتخابی و به منظور تسهیل فرآیند تولید می باشند.

لیست زیر ابزارهای شکل را نشان می دهند:

☐ **C Compiler**: سورس کد C را قبول کرده و سورس کد اسمبلی تولید می کند.

☐ **Assembler**: کد اسمبلی را به زبان ماشین ترجمه می کند. زبان ماشین به فرمت COFF می

باشد.

☐ **Assembly Optimizer**: به شما اجازه می نگرش کد اسمبلی خطی بدون درگیری و نگرانی در

مورد ساختار لوله یا انتخاب ثبات ها را می دهد. این بخش ثبات ها را نصب کرده و بهینه سازی حلقه

ای برای تبدیل اسمبلی خطی به اسمبلی کاملاً موازی که از خط لوله نرم افزار بهره می برد، استفاده می کند.

☐ **Linker**: این بخش فایل های موضوعی را با یک ماژول اجرایی تلفیق می کند.

همگام با تولید ماژول اجرایی عملیات جابجایی و رفع مراجع خارجی نیز انجام می شود.

ورودی لینکر فایل های موضوعی COFF قابل جابجا شدن و کتابخانه های موضوعی می باشد.

☐ **Achiever**: به شما اجازه جمعیت گروهی از فایل ها به یک آرشیو تکی را می دهد که کتابخانه

نامیده می شود. همچنین اجازه می تصحیح کتابخانه به منظور پاک کردن، جابجایی، استخراج و یا اضافه کردن اعضا را فراهم آورده است.

☐ **Library-Built Utility**: شما می توانید از این بخش برای تولید کتابخانه های خود استفاده

کنید.

☐ **Run-Time-Support Library**: حاوی توابع استاندارد ANSI، توابع ریاضی نقطه شناور و

توابع I/O که توسط کامپایلر C پشتیبانی می شود، می باشد.

☐ **Hex-Conversion Utility**: یک فایل موضوعی COFF را به فرمت TI-tagged،

ASCII-hex، Intel، Motorola-s، Tektronix تبدیل می کند. شما می توانید فایل تبدیل

شده را به یک برنامه نویسی EPROM بارگزاری کنید.

☐ **Cross-Reference Lister**: از فایل های موضوعی برای نشانه گذاری Cross-Reference،

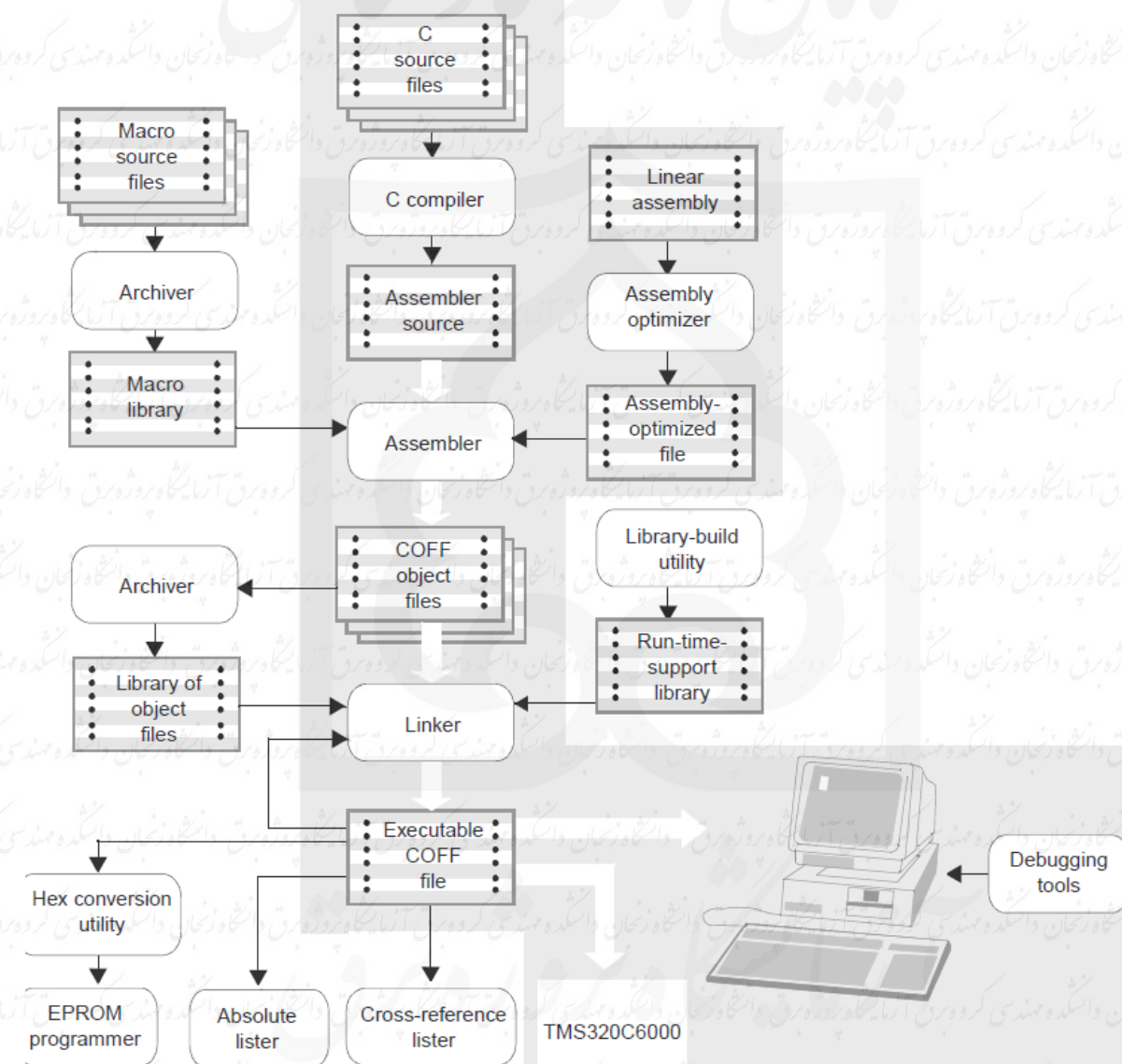
توضیح آنها و سورس آنها در سورس فایل ها استفاده می کند.

Absolute-Lister: فایل های موضوعی مرتبط را به عنوان ورودی قبول می کند و فایل های

x.abs به عنوان خروجی تولید می کند. فایل های abs. برای تولید یک لیست که حاوی آدرس

های مستقیم بیشتری نسبت به آدرس های انتصابی می باشد، استفاده می شود. بدون این قسمت

تولید چنین لیستی طاقت فرسا است و کارهای دستی زیادی نیاز دارد.



۱-۳-۱ ویژگی‌های تصحیح کد برنامه:

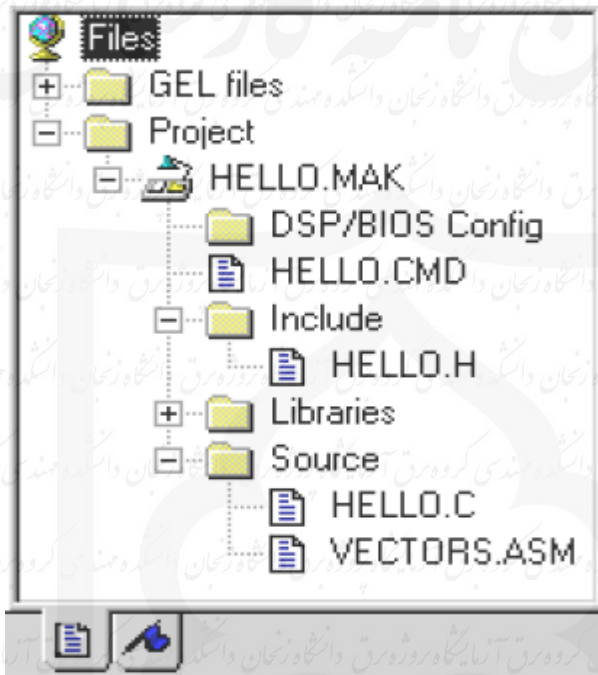
کدنگار به شما اجازه‌ی تصحیح سورس کد C و اسمبلی را فراهم می‌کند. شما هم‌چنین می‌توانید کد C را به همراه اسمبلی نیز مشاهده فرمایید.



- ☐ برجسته کردن کلمات کلیدی، توضیحات و رشته‌ها به صورت رنگی.
- ☐ علامت گذاری بلوک های C در پرانتز ها و براکت ها، پیدا کردن ، تطابق یا انتقال پرانتزها و گروه ها.
- ☐ اضافه و کم کردن فاصله خطوط و مشخص کردن نقطه ی سربرگ.
- ☐ پیدا کردن و جایگزینی یک یا چند فایل، پیدا کردن نمونه قبلی و بعدی و جستجوی سریع.
- ☐ Undo و Redo کردن چند عمل.
- ☐ استفاده از کمک حساس به متن.
- ☐ تطابق، انتخاب‌های، دستورات صفحه کلید.

۱-۳-۲ ویژگی ساخت برنامه کاربردی:

در کدنگار شما می‌توانید یک برنامه را توسط اضافه نمودن فایل‌ها به یک پروژه تولید کنید. فایل پروژه برای ساخت یک برنامه کاربردی استفاده می‌شود. فایل‌های داخل یک پروژه می‌توانند شامل فایل‌های با مرجع C، مرجع اسمبلی، فایل‌های موضوعی، کتابخانه‌ها، فایل‌های دستوری اتصال (Linker Command) و فایل‌های جامع باشند.



شما می‌توانید از یک پنجره برای تصریح گزینه‌های مورد استفاده حین کامپایل، اسمبل و اتصال یک پروژه بهره ببرید.

با استفاده از یک پروژه، کدنگار می‌تواند یک ساخت کلی یا یک ساخت تدریجی تولید کند و همچنین می‌تواند فایل‌های تکی را کامپایل کند. همچنین می‌تواند فایل‌ها را به منظور تولید یک درخت وابستگی برای یک فایل جامع برای کل پروژه تولید کند.

امکانات ساخت کدنگار می‌تواند به عنوان ساخت فایل‌ها به شیوه سنتی نیز استفاده شود.

۱-۳-۳ ویژگی عیب‌یابی برنامه:

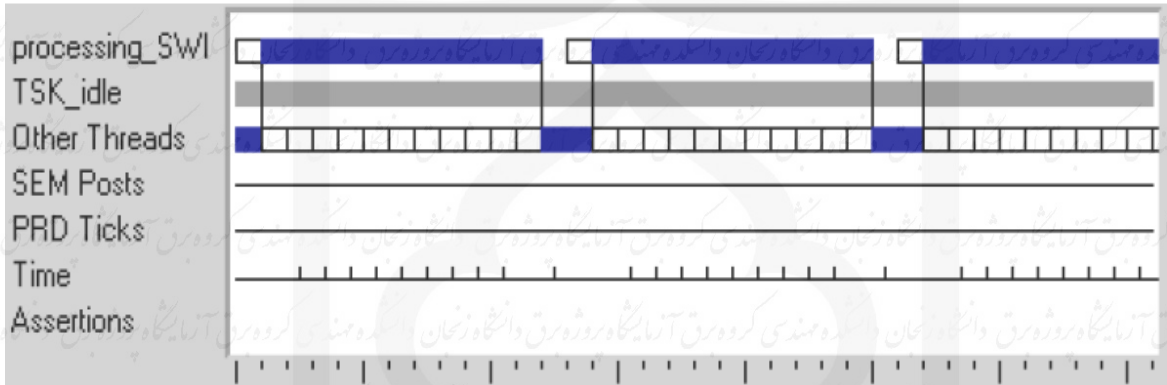
کدنگار از فعالیت‌های عیب‌یابی زیر پشتیبانی می‌کند.

□ ایجاد نقطه شکست (Breaking point) با یک سری از تنظیمات گام.

۱-۴ اتصالات DSP/Bios:

در طول فاز آنالیز توسعه نرم افزار ویژگی های خطایابی سنتی برای تشخیص مشکلات ریز که از فعل و انفعالات وابسته به زمان به وجود می آیند، غیرفعال می باشد.

اتصالات کدنگار توسط پشتیبانی DSP/Bios همچون آنالیزهای بلادرنگ پشتیبانی می شود. شما می -
توانید از آنها برای کاوش بصری، دنباله روی و نظارت بر یک برنامه DSP با کمترین تاثیر روی اجرای
بلادرنگ بهره مند شوید. برای مثال، گراف اجرایی نشان داده شده در شکل زیر یک توالی که در آن
برنامه های مختلف به صورت رشته ای اجرا شدند را نشان می دهد.



DSP/Bios API توانایی های آنالیز بلادرنگ زیر را نشان می دهد:

- ☐ **دنباله روی برنامه (Trace):** نمایش رویداد های نوشته شده در گزارشات مرجع و بازخورد دینامیک کنترل جریان در طول اجرای برنامه.
- ☐ **نظارت بر اجرا:** پیگیری آمار و ارقام که استفاده از منابع مرجع را بازتاب می کند. همچون بارگذاری پردازنده و زمان بندی رشته ای.
- ☐ **جریان سازی فایل:** اتصال فایل های موضوعی I/O متعلق به برد مرجع به فایل های میزبان.
- ☐ **DSP/Bios** یک زمان بند بر پایه اولویت را که می توانید در برنامه استفاده کنید فراهم می آورد.

این زمان بند یک اجرای پریودیک از توابع و رشته های چند اولییتی را نشان می دهد.

۱-۴-۱ پیکربندی DSP/Bios:

شما می‌توانید فایل‌های پیکربندی در محیط کدنगर که فایل‌های موضوعی مورد استفاده در DSP/Bios API را توصیف می‌کند، ایجاد نمایید. این فایل می‌تواند نگاشت حافظه و نگاشت پیکانی سخت افزار ISR را ساده کند و لذا ممکن است این بخش در استفاده از DSP/Bios API مورد استفاده قرار گیرد.

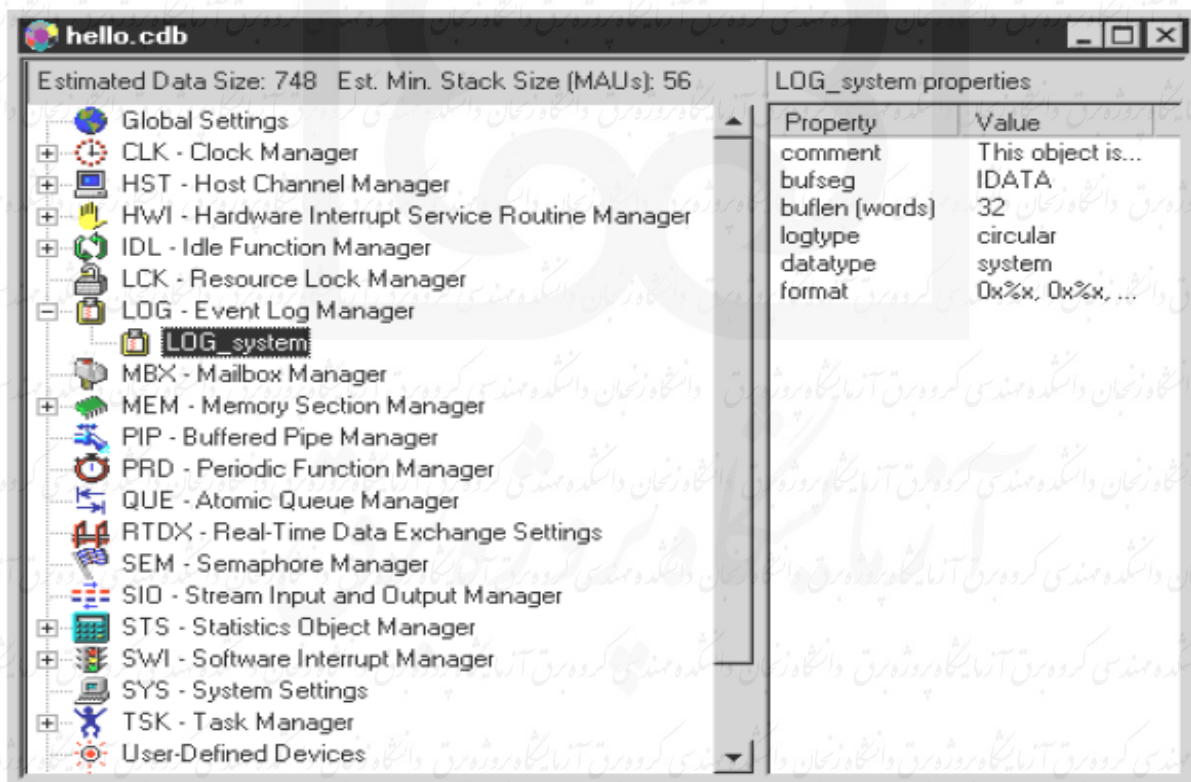
یک فایل پیکربندی دو نقش را ایفا می‌کند.

□ اجازه‌ی استفاده از پارامترهای جامع در هنگام اجرا را فراهم می‌نماید.

□ این فایل‌ها به عنوان یک تصحیح بصری برای تولید و ایجاد ویژگی‌هایی برای فایل‌های موضوعی در حال اجرا که در فراخوانی‌های برد مرجع DSP/Bios API به کار می‌رود، مورد استفاده قرار می‌گیرد.

این فایل‌های موضوعی شامل وقفه‌های نرم‌افزاری، خط‌لوله‌های I/O و گزارشات رویدادها می‌باشد.

اگر شما یک فایل پیکربندی در کدنगर ایجاد کنید یک پنجره همچون شکل زیر ایجاد می‌شود:

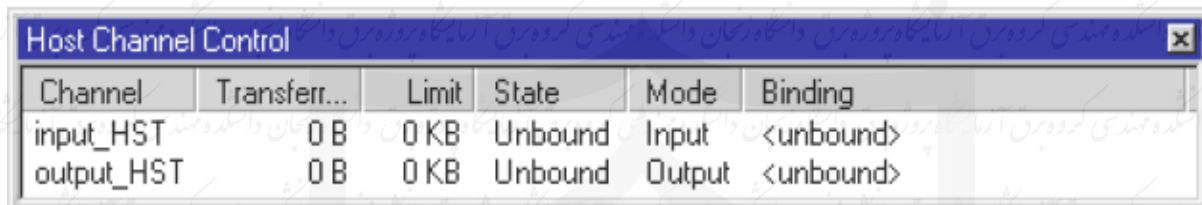


۷-۸ اجرای برنامه اصلاح شده

۱) Project → Rebuild All را انتخاب نموده و یا روی کلید  (بازسازی کلی) در نوار ابزار کلیک نمایید.

۲) File → Load Program را انتخاب نمایید، برنامه‌ی تازه‌بازسازی‌شده را انتخاب نمایید (hostio.out) و آن را باز کنید.

۳) Tools → DSP/Bios → Host Channel را انتخاب نمایید. کنترل کانال میزبان فایل‌های موضوعی HST را لیست می‌نماید و به شما اجازه‌ی اتصال آنها به فایل‌های روی کامپیوتر میزبان و شروع و پایان کانال‌های آن داده می‌شود.



Channel	Transferr...	Limit	State	Mode	Binding
input_HST	0 B	0 KB	Unbound	Input	<unbound>
output_HST	0 B	0 KB	Unbound	Output	<unbound>

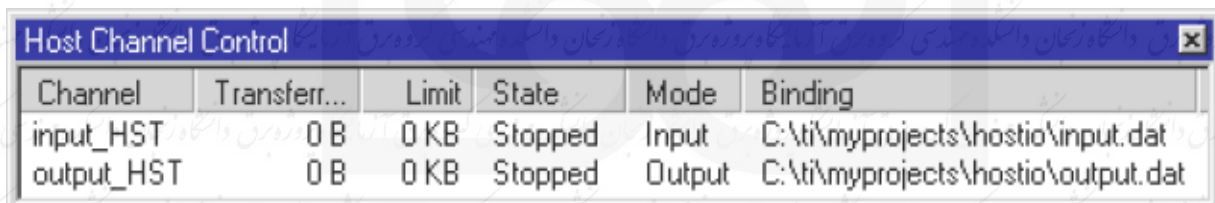
۴) Debug → Run را انتخاب نمایید و یا دکمه‌ی  (اجرا) را فشار دهید.

۵) روی کانال input_HST راست کلیک نمایید و گزینه‌ی Bind را انتخاب کنید.

۶) فایل input.dat را در پوشه‌ی کاری انتخاب کنید و گزینه‌ی Bind را انتخاب کنید.

۷) روی کانال output_HST راست کلیک نمایید و Bind را انتخاب کنید.

۸) Output.dat را در نام فایل تایپ نمایید و Bind را کلیک نمایید.



Channel	Transferr...	Limit	State	Mode	Binding
input_HST	0 B	0 KB	Stopped	Input	C:\ti\myprojects\hostio\input.dat
output_HST	0 B	0 KB	Stopped	Output	C:\ti\myprojects\hostio\output.dat

۹) روی کانال input_HST راست کلیک نمایید و Start را انتخاب کنید.

۱۰) روی کانال output_HST راست کلیک کرده و Start را انتخاب کنید. توجه کنید که ستون

Transferred انتقال داده را نشان می‌دهد.

۱۱) وقتی که داده انتقال یافت دکمه‌ی  (توقف) در نوار ابزار را کلیک نمایید و یا دکمه‌ی shift F5 را

فشار دهید.