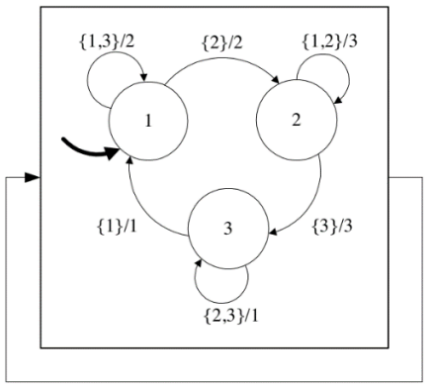
به نام خدا

۱.



الف) ابتدا برای هر استیت وجود fixed point و unique بودن آن را بررسی میکنیم

بررسی استیت ۱:

- Existence: از استیت ۱ با ورودی ۲ خروجی ۲ است. (transition از ۱ به ۲)
- Uniqueness: از استیت ۱ یک transition دیگر هم داریم که ورودی و خروجی برابر نیستند (با ورودی {۱و۳} خروجی ۲ است) پس fixed point نیست.

بنابراین استیت ۱ unique fixed point دارد.

بررسی استیت ۲:

- Existence: از استیت ۲ با ورودی ۳ خروجی ۳ است.(transition از استیت ۲ به ۳)
- Uniqueness: از استیت ۲ یک transition دیگر هم داریم که ورودی و خروجی برابر نیستند(ورودی {۱و۲} و خروجی ۳) پس fixed point نیست.

بنابراین استیت ۲ unique fixed point دارد.

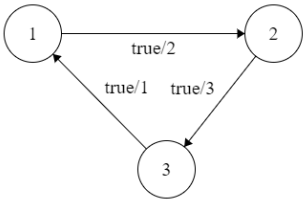
بررسی استیت ۳:

- Existence: از استیت ۳ با ورودی ۱ خروجی ۱ است. (transition از ۳ به ۱)
- Uniqueness: از استیت ۳ یک transition دیگر هم داریم که ورودی و خروجی برابر نیستند(ورودی {۲و۳} و خروجی ۱) پس fixed point نیست.

بنابراین استیت۳ unique fixed point دارد.

پس مدل خوش ساخت است.(با فرض اینکه default transition نداریم زیرا اگر داشته باشیم دیگر در هر استیت شرط unique بودن برقرار نمیشود و مدل دیگر خوش ساخت نمیشود.)

ب) بنابراین می توان ماشین حالت قسمت الف را به این شکل نشان داد:



پس خروجی ۱۰ واکنش اول:

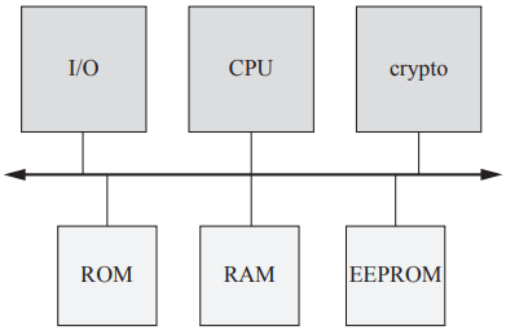
۲-۳-۱-۲-۳-۱-۲-۳-۱-۲ ⟶

ج) بله زیرا در هر استیت اگر ورودی unknown باشد خروجی مشخص است. (در استیت ۱ خروجی ۲ است و میتوان بلافاصله نتیجه گرفت ورودی ۲ است. در استیت ۲ خروجی ۳ است و میتوان بلافاصله نتیجه گرفت ورودی ۳ است و در استیت ۳ خروجی ۱ است و میتوان بلافاصله نتیجه گرفت ورودی ۱ است)

۲.

الف) برنامه ها و منابع سخت افزاری اغلب به عنوان trusted یا untrusted طبقه بندی میشوند. در برنامه trusted امتیازات بیشتری مجاز است: توانایی تغییر مکان‌هاش مشخصی از حافظه، دسترسی به ادوات io و غیره. برنامه‌های untrusted مجاز به اجرای مستقیم برنامه‌های trusted نیستند زیرا اگر مجاز بودند ممکن بود سطح بالاتری از اعتماد را برای خود می گرفتند که این سطح بالاتر به آنها اجازه می داد عملیاتی انجام دهند که مجاز نیستند.سیستم باید بتواند سطح اعتماد یک برنامه را قبل از دادن وضعیت قابل اعتماد به آن برنامه تعیین کند. می توان از امضای دیجیتالی برای برنامه استفاده کرد تا مشخص شود که از یک منبع قابل اعتماد آمده است.با این حال، کلید عمومی مورد استفاده برای بررسی امضای دیجیتال خود باید قابل اعتماد باشد.ما باید مطمئن باشیم که دشمن کلید عمومی را تغییر نداده است تا بتواند امضاها را جعل کند. قابل اعتماد بودن کلید عمومی مستلزم ارزیابی سطح اعتماد منبع آن است که پس از آن باید قابل اعتماد باشد.ما باید مطمئن باشیم که دشمن کلید عمومی را تغییر نداده است تا بتواند امضاها را جعل کند. قابل اعتماد بودن کلید عمومی مستلزم ارزیابی سطح اعتماد منبع آن است، که پس از آن باید قابل اعتماد باشد. در نهایت، ارزیابی اعتماد باید به یک root of trust (ریشه اعتماد) - منبع اصلی نرم افزار قابل اعتماد در سیستم منتهی شود. چندین روش را می توان برای ایجاد یک ریشه اعتماد مورد استفاده قرار داد. یک روش، همانطور که در مثال بعدی توضیح داده شد، جاسازی نرم افزار امضا شده و کلید عمومی مرتبط در سخت افزار غیرقابل تغییر است.

ب) کارت های هوشمند به طور گسترده ای برای تراکنش هایی که شامل پول یا سایر اطلاعات حساس است استفاده می شود. یک تراشه کارت هوشمند باید چندین محدودیت را برآورده کند: باید ذخیره سازی ایمن برای اطلاعات فراهم کند. باید اجازه دهد برخی از آن اطلاعات تغییر کند. باید در سطوح انرژی بسیار پایین کار کند. و باید با هزینه بسیار کم ساخته شود. تصویر زیر معماری یک کارت هوشمند معمولی [NXP14] را نشان می دهد. تراشه کارت هوشمند به گونه ای طراحی شده است که فقط در صورت استفاده از منبع تغذیه خارجی کار کند.



واحد io به تراشه اجازه می دهد تا با یک ترمینال خارجی صحبت کند. هم می توان از کنتاکت های الکتریکی سنتی و هم ارتباطات غیر تماسی استفاده کرد. CPU برای محاسبات به RAM دسترسی دارد اما از حافظه غیر فرار نیز استفاده می کند. یک رام ممکن است برای ذخیره کدی که قابل تغییر نیست استفاده شود. ممکن است کارت بخواهد برخی از داده‌ها یا برنامه‌ها را تغییر دهد و آن مقادیر را حتی در صورت عدم استفاده از برق حفظ کند. یک رام قابل برنامه ریزی با قابلیت پاک شدن الکتریکی (EEPROM) اغلب برای این حافظه غیر فرار به دلیل هزینه بسیار پایین آن استفاده می شود. مدار تخصصی استفاده می شود تا به CPU اجازه می دهد تا به EEPROM بنویسد تا اطمینان حاصل شود که سیگنال های

نوشتن حتی در طول عملیات CPU پایدار هستند [Ugo86]. یک واحد رمزنگاری، همراه با یک کلید که ممکن است در ROM یا سایر حافظه های دائمی ذخیره شود، رمزگذاری و رمزگشایی را فراهم می کند.

ج) ARM TrustZone [ARM09] به ماشین‌ها اجازه می‌دهد تا با واحدهای زیادی طراحی شوند که می‌توانند در یکی از دو حالت عادی یا ایمن کار کنند. پردازنده‌های دارای TrustZone یک بیت وضعیت NS دارند که تعیین می‌کند در حالت امن یا عادی کار کند. گذرگاه‌ها، کنترل‌کننده‌های DMA و کنترل‌کننده‌های کش نیز می‌توانند در حالت امن کار کنند.

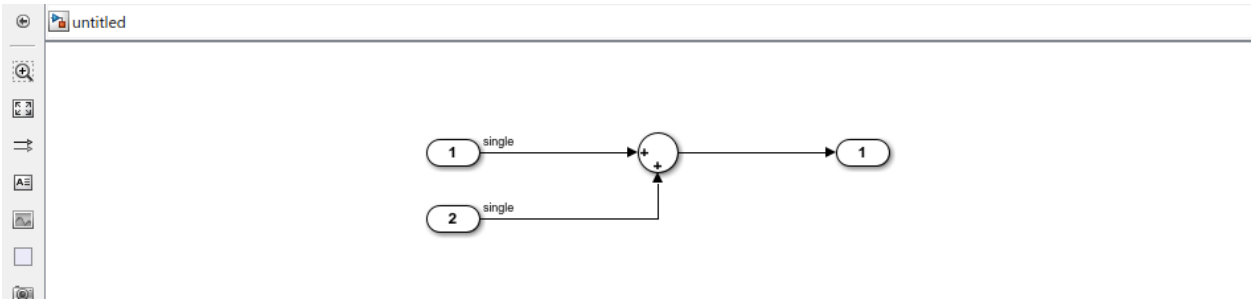
۳. ویژگیهای لایه‌های NWK و APL پروتکل Zigbee
ZigBee [Far08] دو لایه را بالای لایه های PHY و MAC 802.15.4 تعریف می کند: لایه NWK خدمات شبکه را ارائه می دهد و لایه APL خدمات در سطح برنامه را ارائه می دهد.
لایه ZigBee NWK سرویس‌های شبکه را ارایه می‌دهد، ورود و خروج دستگاه‌ها را به شبکه و از شبکه مدیریت می‌کند و مسیریابی را مدیریت می‌کند. لایه NWK دو جزء اصلی دارد: نهاد داده لایه NWK (NLDE) خدمات انتقال داده را ارائه می دهد. نهاد مدیریت لایه NWK (NLME) خدمات مدیریتی را ارائه می دهد. یک پایگاه اطلاعات شبکه (NIB) مجموعه ای از ثابت ها و ویژگی ها را در خود جای داده است. لایه NWK همچنین یک آدرس شبکه برای دستگاه تعریف می کند.
لایه NWK سه نوع ارتباط را فراهم می کند: پخش، چندپخشی و یونیکست.
یک پیام پخش توسط هر دستگاه در کانال پخش دریافت می شود. پیام های چند پخشی به مجموعه ای از دستگاه ها ارسال می شوند. یک پیام unicast، نوع پیش‌فرض ارتباط، به یک دستگاه ارسال می‌شود.
به طور کلی یک پیام ممکن است از طریق چندین پرش در شبکه به مقصد خود برسد. هماهنگ کننده یا روتر Zig Bee یک فرآیند مسیریابی را برای تعیین مسیر از طریق شبکه ای که برای برقراری ارتباط با یک دستگاه استفاده می شود، انجام می دهد. انتخاب مسیر را می توان با عوامل متعددی هدایت کرد: تعداد پرش یا کیفیت پیوند.لایه NWK تعداد پرش هایی را که یک فریم معین مجاز به حرکت است، محدود می کند.

لایه ZigBee APL شامل یک چارچوب برنامه کاربردی، یک زیر لایه پشتیبانی برنامه (APS) و یک شی دستگاه ZigBee (ZDO) است. چندین شی برنامه ممکن است توسط چارچوب برنامه مدیریت شوند که هر کدام برای یک برنامه متفاوت است. APS رابط خدماتی را از لایه NWK به اشیاء برنامه ارائه می دهد. ZigBee Device Object رابط های اضافی بین APS و چارچوب برنامه فراهم می کند.

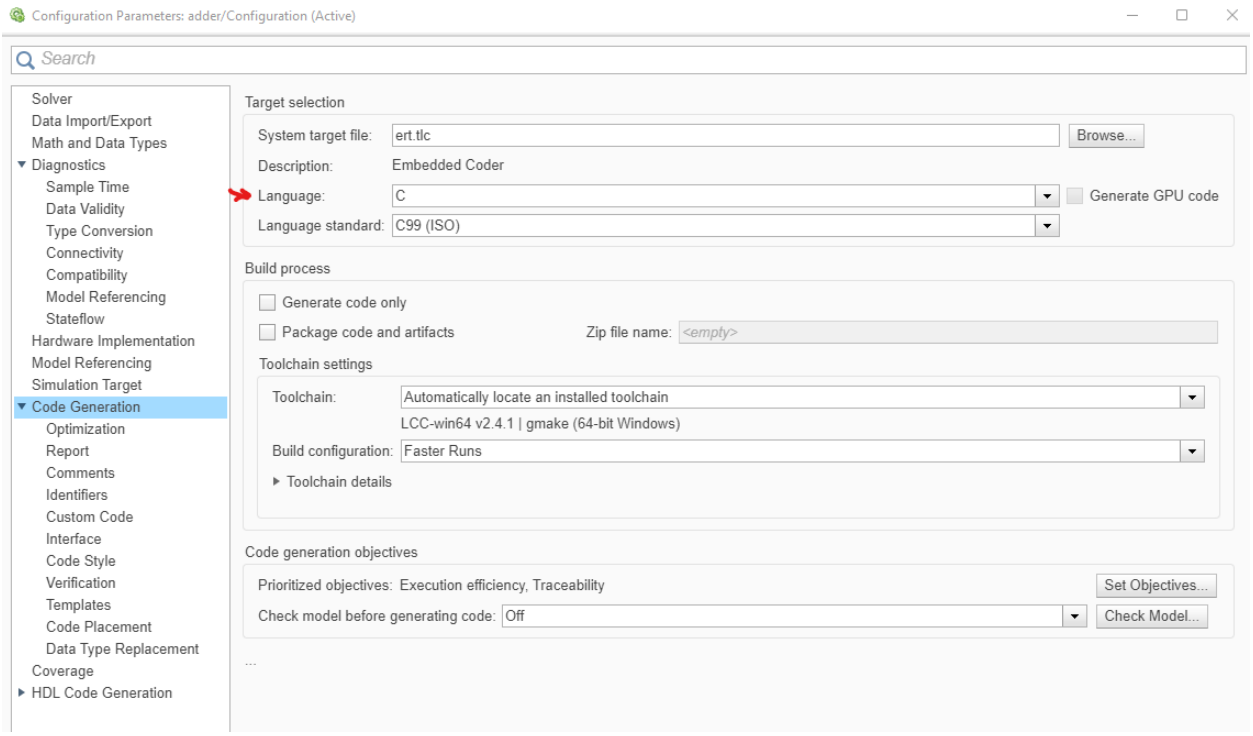
ZigBee تعدادی پروفایل برنامه را تعریف می کند که یک برنامه خاص را تعریف می کند. شناسه برنامه توسط ZigBee Alliance صادر می شود. نمایه برنامه شامل مجموعه ای از توضیحات دستگاه است که ویژگی ها و وضعیت دستگاه را نشان می دهد. یکی از عناصر توضیحات دستگاه نیز به خوشه ای اشاره می کند که از مجموعه ای از ویژگی ها و دستورات تشکیل شده است.

۴.
الف) ابتدا سراغ simulink میرویم و با صفحه زیر مواجه میشویم:

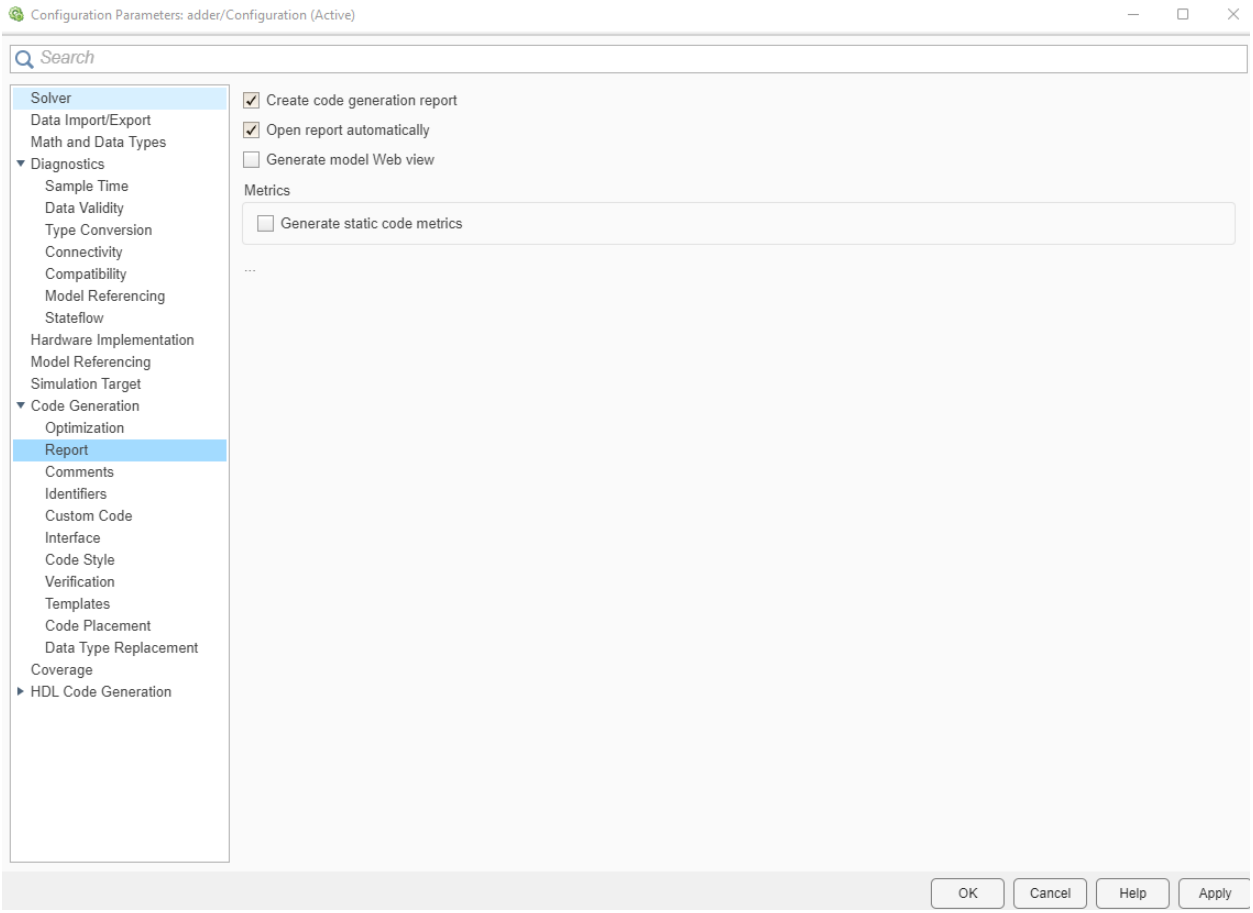
از بین موارد بالا مورد آخر یعنی fixed-step, single-rate را طبق صورت سوال انتخاب میکنیم. پس از انتخاب آن این صفحه شامل یک جمع کننده را داریم:



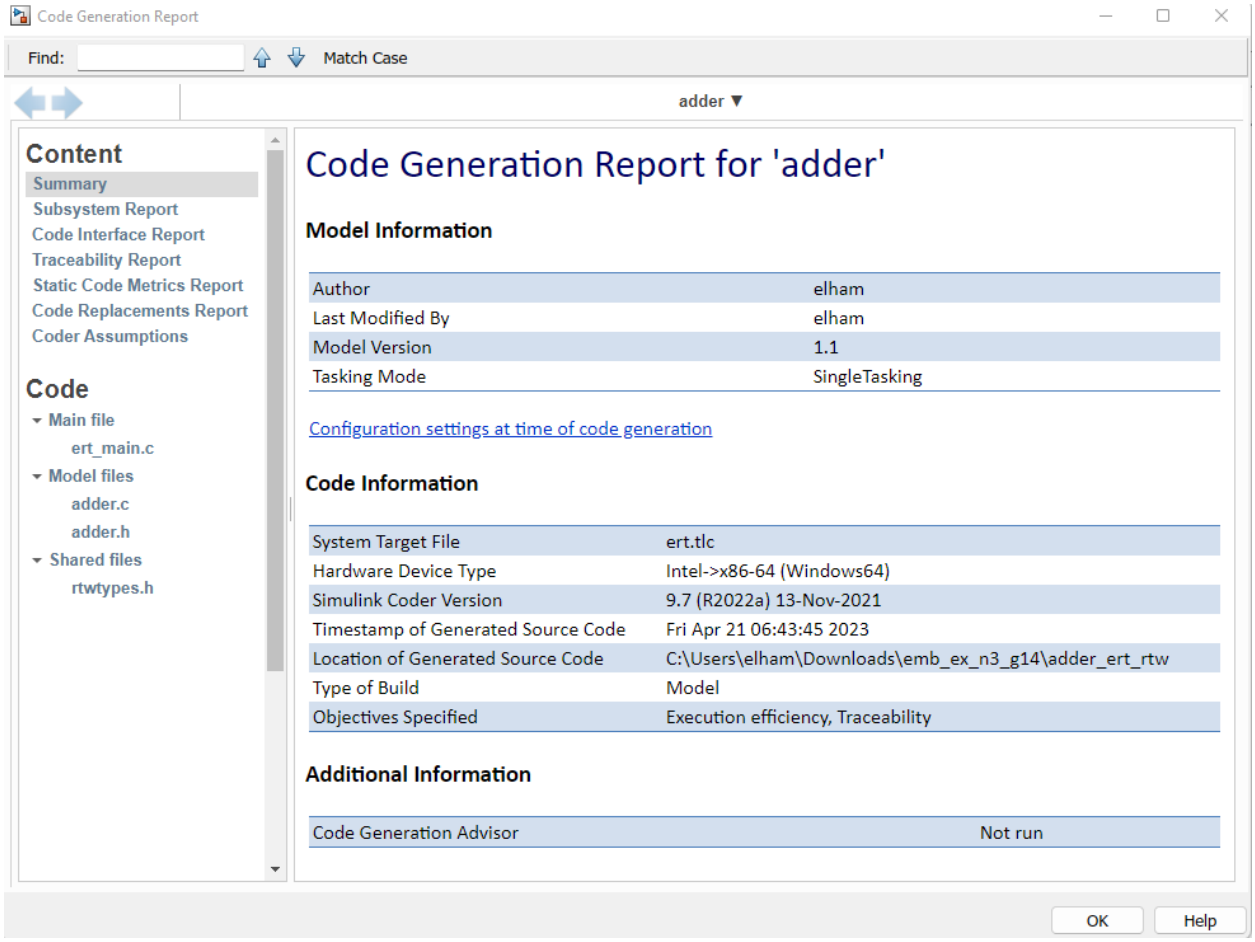
برای تولید کد C به قسمت modeling و سپس بخش تنظیمات میرویم. در قسمت code generation میتوان زبان کد تولیدی را انتخاب کرد:

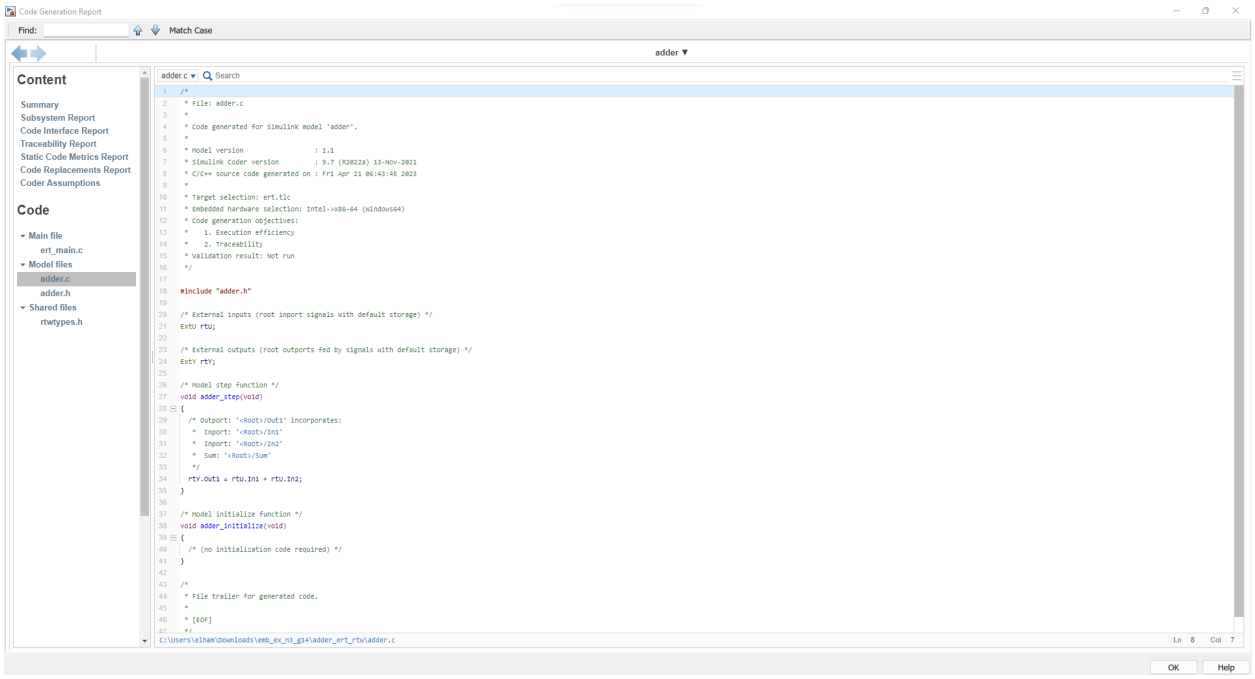


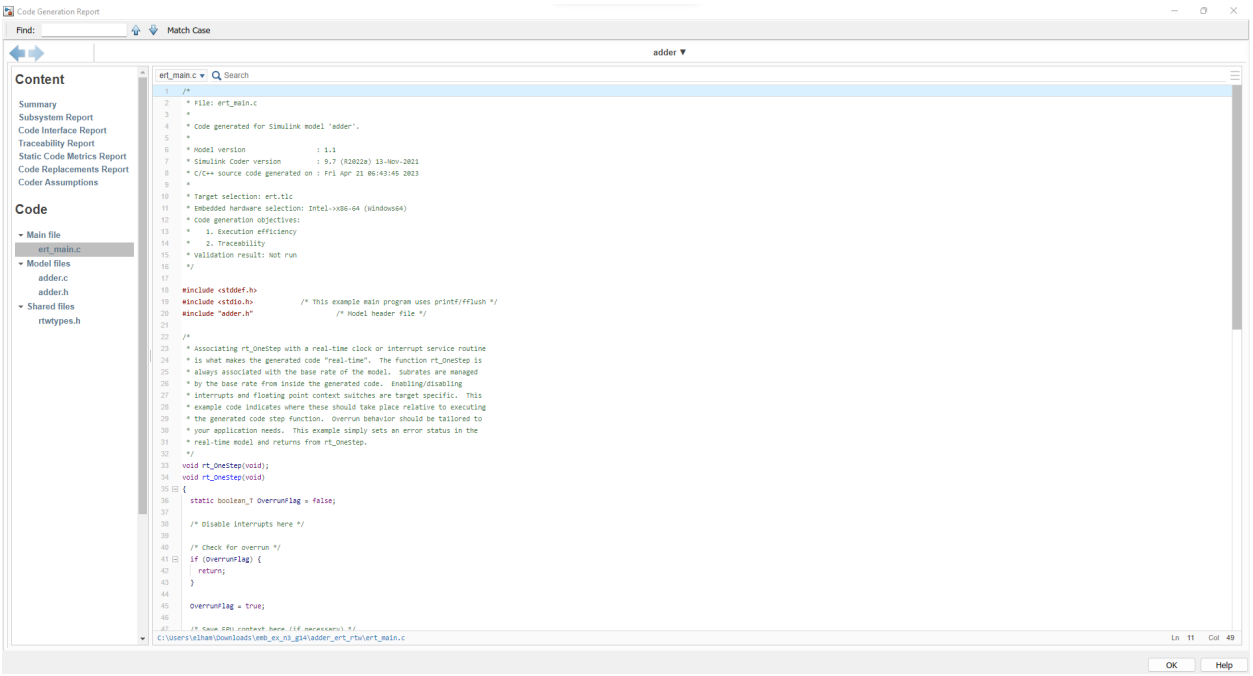
سپس به قسمت report می‌رویم و دو گزینه اول را تیک میزنیم :



در تب apps روی embedded coder کلیک میکنیم تا تب c code باز شود. در این تب با کلیک بر روی build کد c تولید میشود و report هم نشان داده میشود:


Code Generation Report
Find:
Match Case
adder ▼
Content
Summary
Subsystem Report
Code Interface Report
Traceability Report
Static Code Metrics Report
Code Replacements Report
Coder Assumptions
Code
Main file
ert_main.c
Model files
adder.c
adder.h
Shared files
rtwtypes.h
Code Generation Report for 'adder'
Model Information
Author elham
Last Modified By elham
Model Version 1.1
Tasking Mode SingleTasking
Configuration settings at time of code generation
Code Information
System Target File ert.tlc
Hardware Device Type Intel->x86-64 (Windows64)
Simulink Coder Version 9.7 (R2022a) 13-Nov-2021
Timestamp of Generated Source Code Fri Apr 21 06:43:45 2023
Location of Generated Source Code C:\Users\elham\Downloads\emb_ex_n3_g14\adder_ert_rtw
Type of Build Model
Objectives Specified Execution efficiency, Traceability
Additional Information
Code Generation Advisor Not run
OK
Help


Code Generation Report
Find:
Match Case
adder ▼
Content
Summary
Subsystem Report
Code Interface Report
Traceability Report
Static Code Metrics Report
Code Replacements Report
Coder Assumptions
Code
Main file
ert_main.c
Model files
adder.c
adder.h
Shared files
rtwtypes.h
adder.c
Search
/*
* File: adder.c
*
* Code generated for Simulink model 'adder'.
*
* Model version : 1.1
* Simulink Coder version : 9.7 (R2022a) 13-Nov-2021
* C/C++ source code generated on : Fri Apr 21 06:43:45 2023
*
* Target selection: ert.tlc
* Embedded hardware selection: Intel->x86-64 (Windows64)
* Code generation objectives:
* 1. Execution efficiency
* 2. Traceability
* Validation result: Not run
*/
#include "adder.h"
/* External inputs (root inport signals with default storage) */
ExtU rtU;
/* External outputs (root outports fed by signals with default storage) */
ExtY rtY;
/* Model step function */
void adder_step(void)
{
/* Outport: '<Root>/Out1' incorporates:
* Inport: '<Root>/In1'
* Inport: '<Root>/In2'
* Sum: '<Root>/Sum'
*/
rtY.Out1 = rtU.In1 + rtU.In2;
}
/* Model initialize function */
void adder_initialize(void)
{
/* (no initialization code required) */
}
/*
* File trailer for generated code.
*
* [EOF]
*/
C:\Users\elham\Downloads\emb_ex_n3_g14\adder_ert_rtw\adder.c
Ln 8 Col 7
OK
Help



همان‌طور که میبینیم این کد شامل ۴ فایل است: ۲ هدر و ۲ فایل c. در بخش model files دوفایل هدر و c میبینیم که در اصل کد این مدلی هستند که ما طراحی کردیم. Ert_main کد main ما است و ساختارش خیلی شبیه به همان کدهای امبددی است که تا الان میزدیم. کد main شامل یک بخش initialization است:

```
/* Initialize model */
adder_initialize();
```

که مقداردهی‌های اولیه مدل ما را انجام میدهد بر اساس چیزهایی که خودمون مشخص کردیم. همچنین در کامنت‌های کد نوشته شده است که پریود ۰.۲ ثانیه است(با کلاک سیستم کار میکند) اگر تابع onestep را طبق پریود کلاک فراخوانی کنیم مثل این است که مدل یک بار اجرا میشود و خروجی ها به ما داده میشود.

بسته پشتیبانی Simulink برای آردوینو کد بلادرنگ تولید نمی کند.به طور کلی، از حالت external می توان برای نظارت بر داده های سیگنالی که از آردوینو می آید استفاده کرد، زیرا کد تولید شده برای مدل روی برد اجرا می شود. در حالت external می‌توان سیگنال‌ها را با بلوک «To Workspace» به فضای کاری پایه وارد کرد، همانطور که در پیوند مستندات زیر توضیح داده شده است:

https://www.mathworks.com/matlabcentral/answers/358575-how-to-use-simulink-and-arduino-real-time-communication

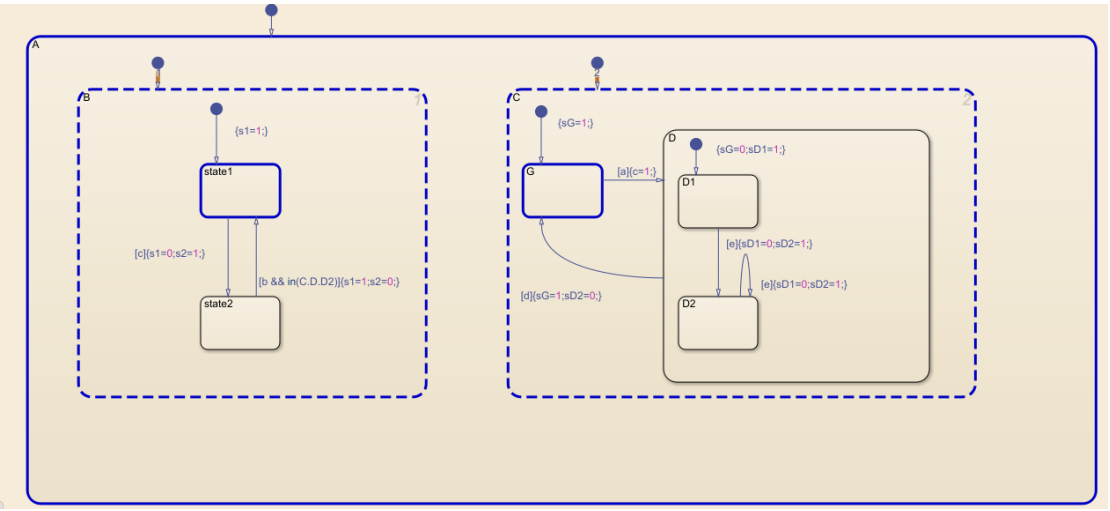
منبع

هر چند که با اجرای تابع onestep در یک لوپ بینهایت میتوان مدل را در فواصل زمانی کم پیوسته اجرا نمود.

```
/* Attach rt_OneStep to a timer or interrupt service routine with
 * period 0.2 seconds (base rate of the model) here.
 * The call syntax for rt_OneStep is
 *
 *  rt_OneStep();
 */
```

ب) ابتدا لازم است مطابق آنچه از ما خواسته شده است مدل را تغییر دهیم. خروجی های s1,s2,sG,sD1,sD2 به همین منظور تعریف و اضافه شده‌اند. وقتی وارد استیت ۱ میشویم s1=1 میشود و وقتی از آن خارج میشویم برابر ۰ میشود. به همین ترتیب برای باقی خروجی ها هم داریم.

در ابتدا وقتی ران میکنیم وارد استیت ۱ و G میشویم و میبینیم به درستی استیت های مربوطه ۱ شدند. داریم:

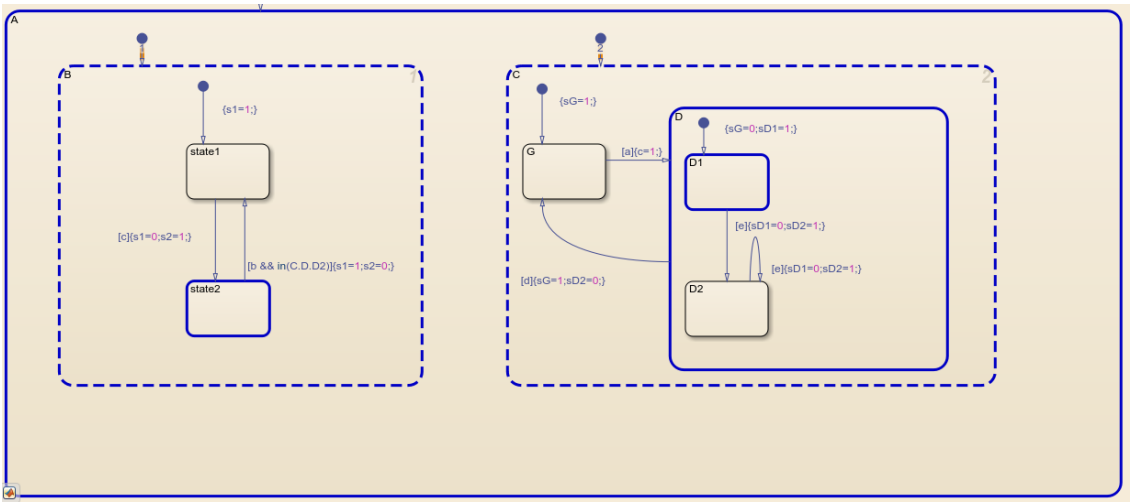


Symbols

filter

TYPE	NAME	VALUE	PORT
	a	0	1
	b	0	2
	e	0	3
	d	0	4
	c	0	
	s2	0	
	s1	1	
	sG	1	
	sD1	0	
	sD2	0	
	A		

سپس وارد استیت D1 و ۲ میشویم و میبینیم متغیرهای مربوطه به درستی آپدیت شدند:

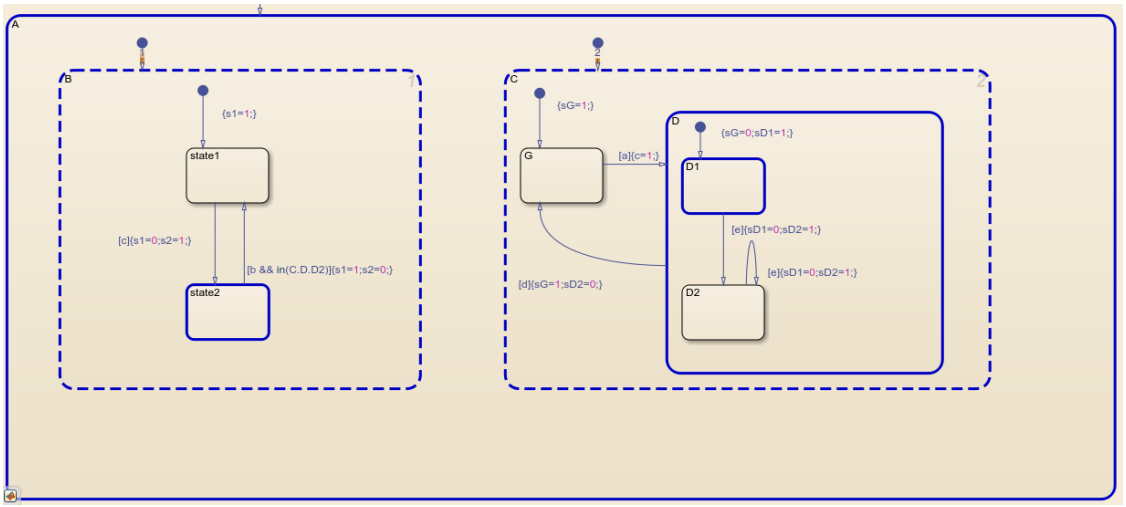


Symbols

filter

TYPE	NAME	VALUE	PORT
	a	1	1
	b	0	2
	e	0	3
	d	0	4
	c	1	
	s2	1	
	s1	0	
	sG	0	
	sD1	1	
	sD2	0	
	A		

سپس b=1 میشود که استیت را عوض نمیکند.

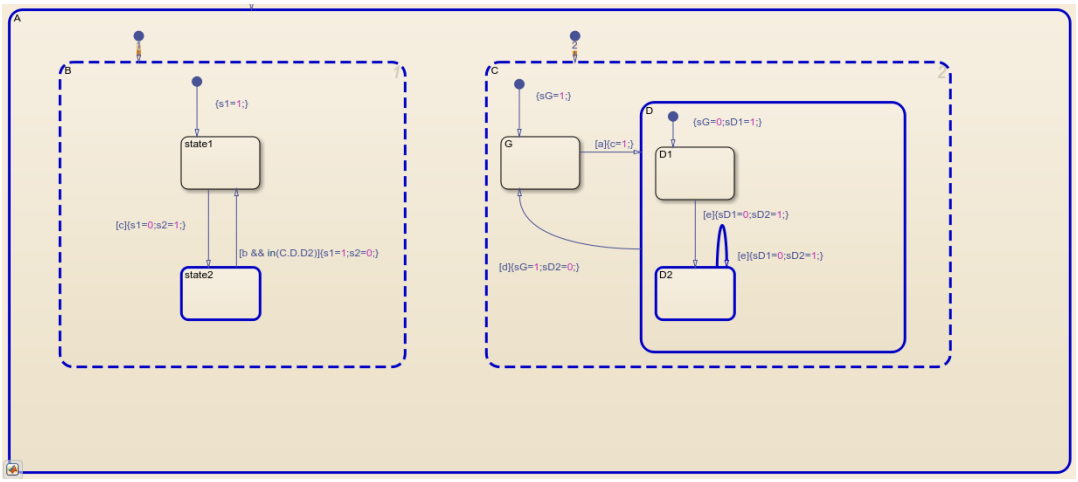


Symbols

filter

TYPE	NAME	VALUE	PORT
	a	0	1
	b	1	2
	e	0	3
	d	0	4
	c	1	
	s2	1	
	s1	0	
	sG	0	
	sD1	1	
	sD2	0	
	A		

سپس وارد استیت D2 میشویم و میبینیم متغیرهای مربوطه به درستی آپدیت شده‌اند.

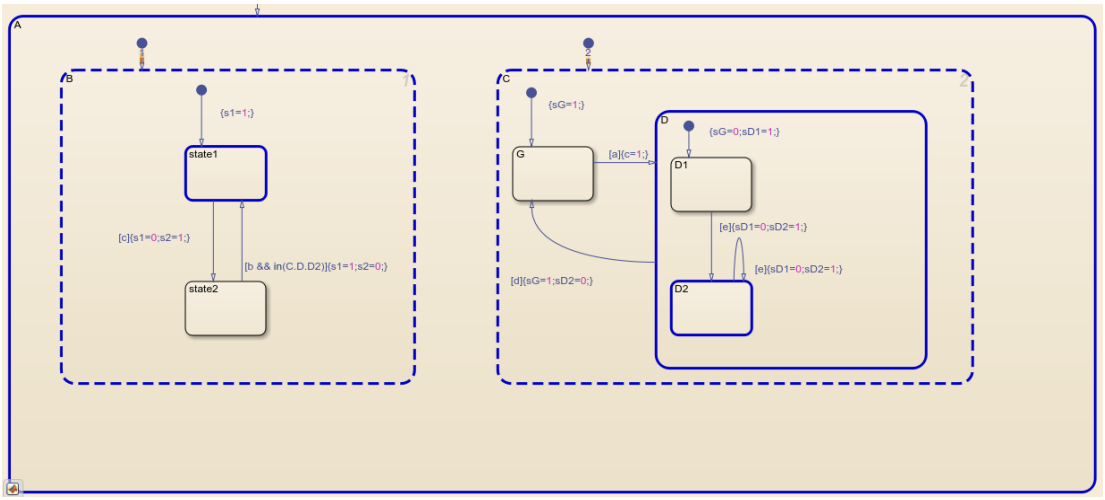


Symbols

filter

TYPE	NAME	VALUE	PORT
	a	0	1
	b	0	2
	e	1	3
	d	0	4
	c	1	
	s2	1	
	s1	0	
	sG	0	
	sD1	0	
	sD2	1	
	A		

بعد با ۱ شدن b وارد استیت ۱ میشویم و متغیرهای مربوطه به درستی آپدیت میشوند:

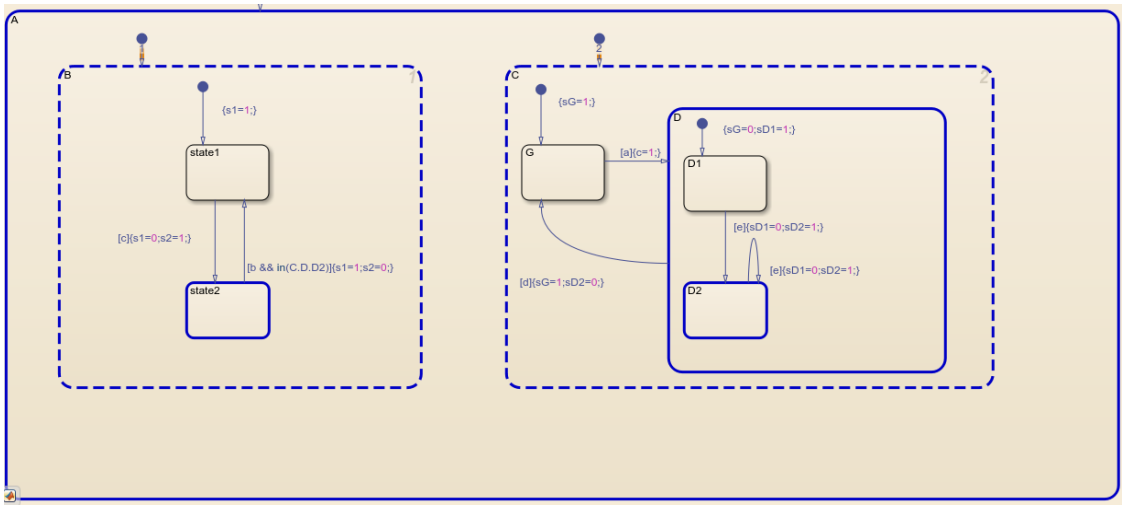


Symbols

filter

TYPE	NAME	VALUE	PORT
	a	0	1
	b	1	2
	e	0	3
	d	0	4
	c	1	
	s2	0	
	s1	1	
	sG	0	
	sD1	0	
	sD2	1	
	A		

در ادامه مدام بین استیت ۱ و۲ جابجا میشود و در نتیجه s1,s2 هم ۰و۱ میشوند.

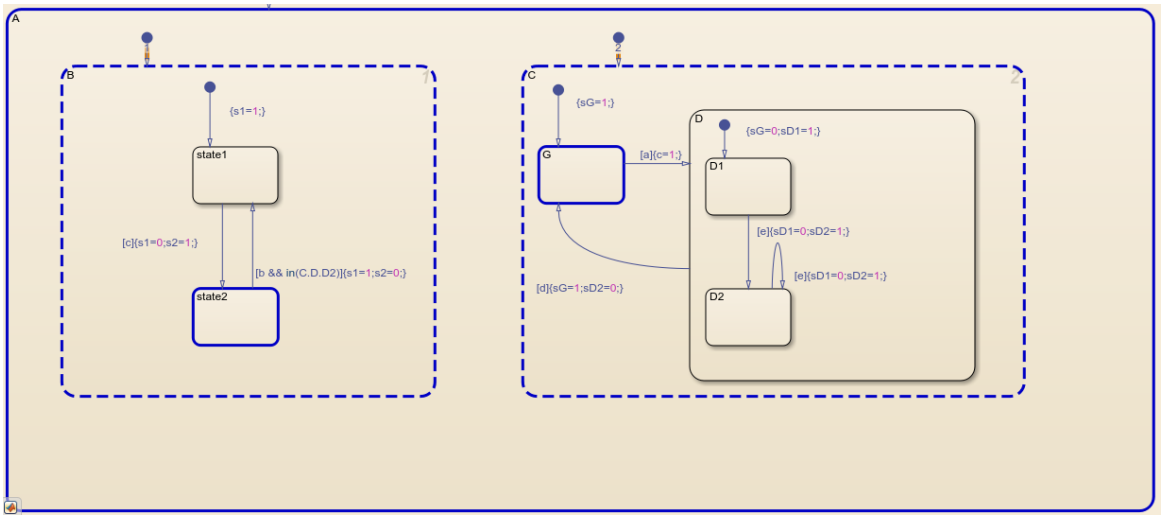


Symbols

filter

TYPE	NAME	VALUE	PORT
	a	0	1
	b	1	2
	e	0	3
	d	0	4
	c	1	
	s2	1	
	s1	0	
	sG	0	
	sD1	0	
	sD2	1	
	A		

در ادامه با ۱ شدن d وارد استیت G میشویم و میبینیم sG=1 شده است:



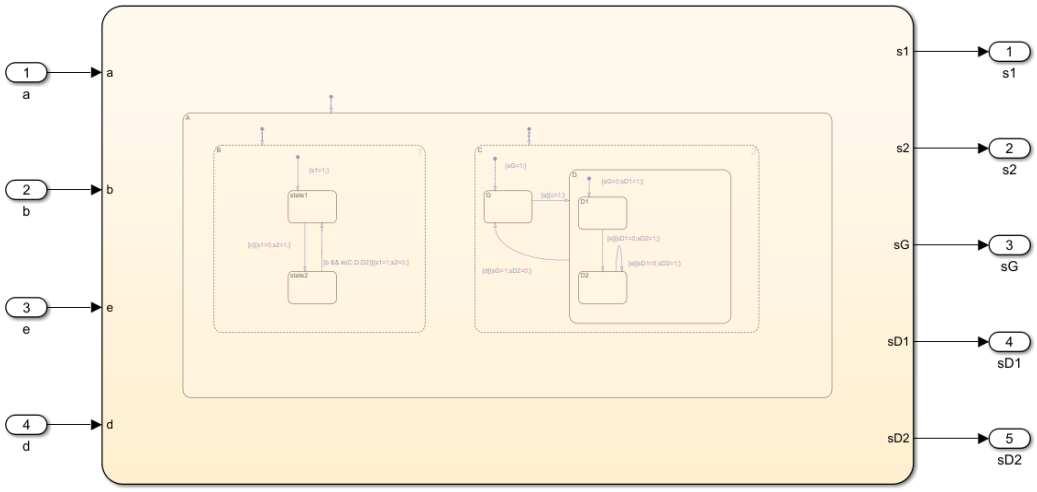
Symbols

filter

TYPE	NAME	VALUE	PORT
	a	0	1
	b	0	2
	e	0	3
	d	1	4
	c	1	
	s2	1	
	s1	0	
	sG	1	
	sD1	0	
	sD2	0	
	A		

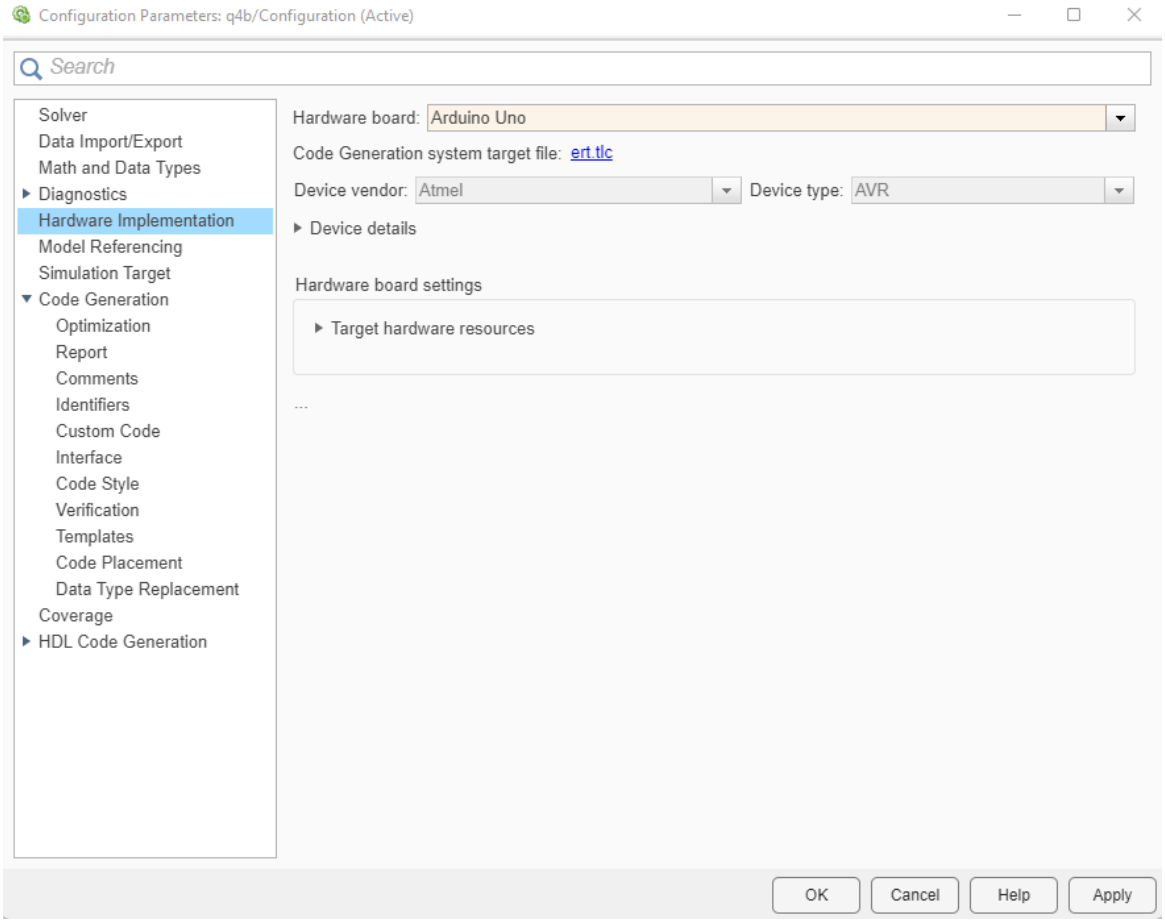
به این ترتیب درستی تغییر ایجاد شده در مدل چک شد.

سپس لازم است برای استیت چارت اینپوت و اوت پوت به شکل خواسته شده تعریف کنیم:



سپس به قسمت apps رفته و embedded coder را انتخاب میکنیم و در تب c code وارد قسمت تنظیمات می شویم.

از منوی سمت چپ hardware implementation را انتخاب میکنیم و board را روی arduino uno میگذاریم.



و در نهایت کد تولید میشود. فایل‌های مختلفی وجود دارد برای مثال تصویر بخشی از کدهای فایل q4b.c را میبینیم:
اجزای کد:
در فایل q4b_c دو استراکت برای ورودی و خروجی تعریف شده است.

```
/* External inputs (root inport signa
ExtU_q4b_T q4b_U;

/* External outputs (root outports fe
ExtY_q4b_T q4b_Y;
```

همچنین در تابع step مقداردهی به خروجی‌ها طبق شرطها انجام شده است.

```
if (q4b_DW.is_active_c3_q4b == 0U) {
  q4b_DW.is_active_c3_q4b = 1U;

  /* Outport: '<Root>/s1' */
  q4b_Y.s1 = 1.0;
  q4b_DW.is_B = q4b_IN_state1;

  /* Outport: '<Root>/sG' */
  q4b_Y.sG = 1.0;
  q4b_DW.is_C = q4b_IN_G;
} else {
```

در فایل ert_main.c ابتدا توابع مقداردهی اولیه فراخوانی شده‌اند سپس در یک حلقه تا زمانی که stoprequest نشود True لوپ arduino اجرا میشود.
در دیگر فایل‌ها برای مثال در فایل rwt_types میبینیم که تایپ‌ها تعریف شده‌اند.

```c
#include "q4b.h"
#include "rtwtypes.h"

/* Named constants for Chart: '<Root>/Chart' */
#define q4b_IN_D                       ((uint8_T)1U)
#define q4b_IN_D1                      ((uint8_T)1U)
#define q4b_IN_D2                      ((uint8_T)2U)
#define q4b_IN_G                       ((uint8_T)2U)
#define q4b_IN_NO_ACTIVE_CHILD         ((uint8_T)0U)
#define q4b_IN_state1                  ((uint8_T)1U)
#define q4b_IN_state2                  ((uint8_T)2U)

/* Block states (default storage) */
DW_q4b_T q4b_DW;

/* External inputs (root inport signals with default storage) */
ExtU_q4b_T q4b_U;

/* External outputs (root outports fed by signals with default storage) */
ExtY_q4b_T q4b_Y;

/* Real-time model */
static RT_MODEL_q4b_T q4b_M_;
RT_MODEL_q4b_T *const q4b_M = &q4b_M_;

/* Model step function */
void q4b_step(void)
{
  /* Chart: '<Root>/Chart' incorporates:
   *  Inport: '<Root>/a'
   *  Inport: '<Root>/b'
   *  Inport: '<Root>/d'
   *  Inport: '<Root>/e'
   */
  if (q4b_DW.is_active_c3_q4b == 0U) {
    q4b_DW.is_active_c3_q4b = 1U;

    /* Outport: '<Root>/s1' */
    q4b_Y.s1 = 1.0;
```

ج)راه های مختلفی برای دریافت ورودی و خروجی مدل با استفاده از کد تولید شده توسط متلب برای آردوینو وجود دارد. برخی از روش های رایج عبارتند از:

1. ارتباط سریال: کد آردوینو را می توان برای دریافت داده های ورودی از پورت سریال و ارسال داده های خروجی از طریق همان پورت برنامه ریزی کرد. سپس متلب می تواند با برد آردوینو با استفاده از رابط سریال برای ارسال داده های ورودی و دریافت داده های خروجی ارتباط برقرار کند.

2. ورودی های آنالوگ و دیجیتال: برد آردوینو دارای چندین پایه ورودی آنالوگ و دیجیتال است که می توان از آنها برای اتصال سنسورها یا سایر دستگاه های ورودی استفاده کرد. کد MATLAB را می توان طوری برنامه ریزی کرد که مقادیر ورودی را از این پین ها بخواند و از آنها به عنوان ورودی مدل استفاده کند.

3. خروجی های PWM: برد آردوینو چندین پایه خروجی PWM (Pulse Width Modulation) نیز دارد که می توان از آنها برای کنترل موتورها یا سایر دستگاه های خروجی استفاده کرد. کد MATLAB را می توان طوری برنامه ریزی کرد که سیگنال های PWM را به این پین ها ارسال کند تا خروجی مدل را کنترل کند.

4. ارتباط I2C یا SPI: برد آردوینو از پروتکل های ارتباطی I2C و SPI نیز پشتیبانی می کند که می توان از آنها برای برقراری ارتباط با دستگاه های دیگر مانند سنسورها یا نمایشگرها استفاده کرد. کد MATLAB را می توان طوری برنامه ریزی کرد که از طریق برد آردوینو با این دستگاه ها ارتباط برقرار کند تا داده های ورودی را دریافت کند یا داده های خروجی را نمایش دهد.

به طور کلی، روش خاصی که برای دریافت ورودی و خروجی مدل استفاده می شود، به نیازهای خاص برنامه و اجزای سخت افزاری مورد استفاده بستگی دارد.
بسته پشتیبانی MATLAB برای آردوینو لیستی از توابع را ارائه می دهد که امکان دسترسی به ورودی ها و خروجی های دیجیتال و آنالوگ را فراهم می کند. • writeDigitalPin: روی یک خروجی دیجیتال می نویسد. • readDigitalPin: یک ورودی دیجیتال را می خواند. •writePWMDutyCycle: در یک خروجی PWM مینویسد duty cycle در 0-1 • WritePWMVoltage: در خروجی PWM می نویسد، duty cycle در 0-5 (ولت موثر). • readVoltage: ورودی آنالوگ را می خواند.

د) لازم است ابتدا سراغ فایل q4b.c برویم و تمام include ها را با کد آنها جایگزین کنیم. سپس کد را در بخش code سایت tinkercad می گذاریم و دو تابع setup و loop را کامل میکنیم. تابع initialize را در setup و تابع step را در لوپ فراخوانی میکنیم.
در مدار زیر مشخص است که پین های ۲و۳و۴و۵ برای ورودی در نظر گرفته شده است.
در کد نیز داریم:

```
int a = 2;
int b = 3;
int d = 4;
int e = 5;
```

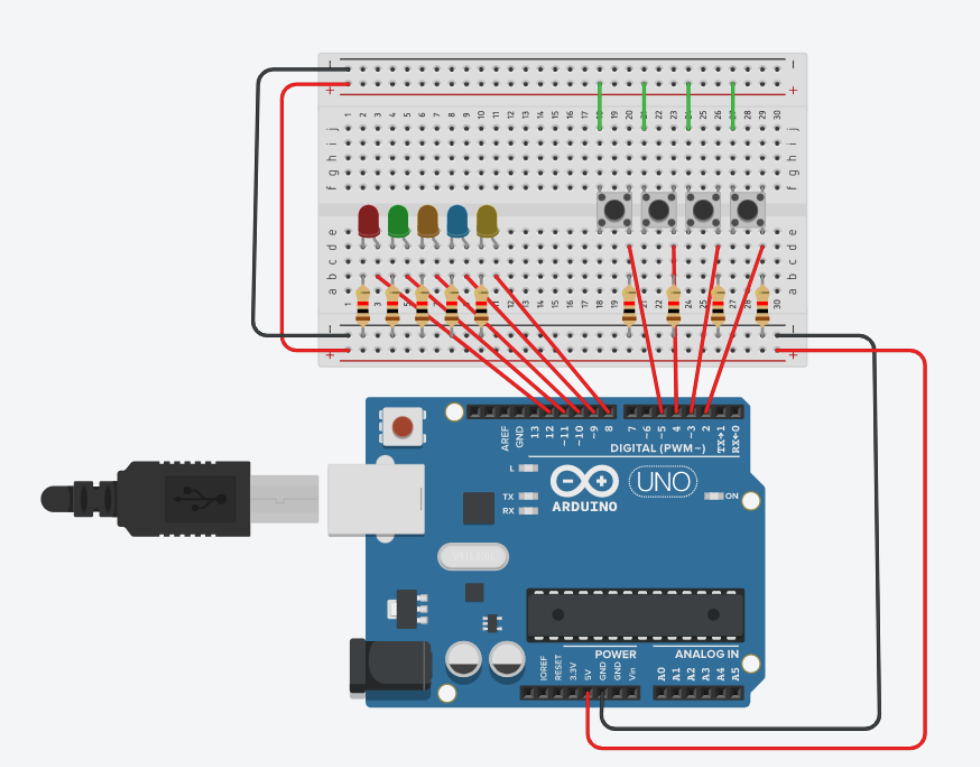
```
pinMode(a, INPUT);
pinMode(b, INPUT);
pinMode(d, INPUT);
pinMode(e, INPUT);
```

همچنین پین‌های ۸و۹و۱۰و۱۱و۱۲ برای خروجی در نظر گرفته شده است.
در کد نیز داریم:

```
int s1 = 8;
int s2 = 9;
int sG = 10;
int sD1 = 11;
int sD2 = 12;
```

```
pinMode(s1, OUTPUT);
pinMode(s2, OUTPUT);
pinMode(sG, OUTPUT);
pinMode(sD1, OUTPUT);
pinMode(sD2, OUTPUT);
```

q4b_U استراکت ورودی و q4b_Y استراکت خروجی ما هستند و از توابع digitalRead و digitalWrite آردوینو برای خواندن و نوشتن ورودی و خروجی استفاده می کنیم.
در ادامه طرح مدار و لینک آن را مشاهده میکنید.



لینک:
https://www.tinkercad.com/things/gbExf3XuCj5-magnificent-curcan-elzing/editel?tenant=circuits

تست: ابتدا باید وارد s1 و sG بشویم که چراغ های مربوطه به درستی روشن میشوند.
بقیه حالات هم به همین ترتیب درست است.