

۲. هدف این بخش تمرین، استفاده از ابزار Fixed-Point Designer برای جایگزینی نوع‌های ممیز شناور با نوع‌های ممیز ثابت مناسب در مدل روبات توسعه داده شده تمرین ۵ و مقایسه زمان اجرای کدهای تولید شده از آن‌ها است.

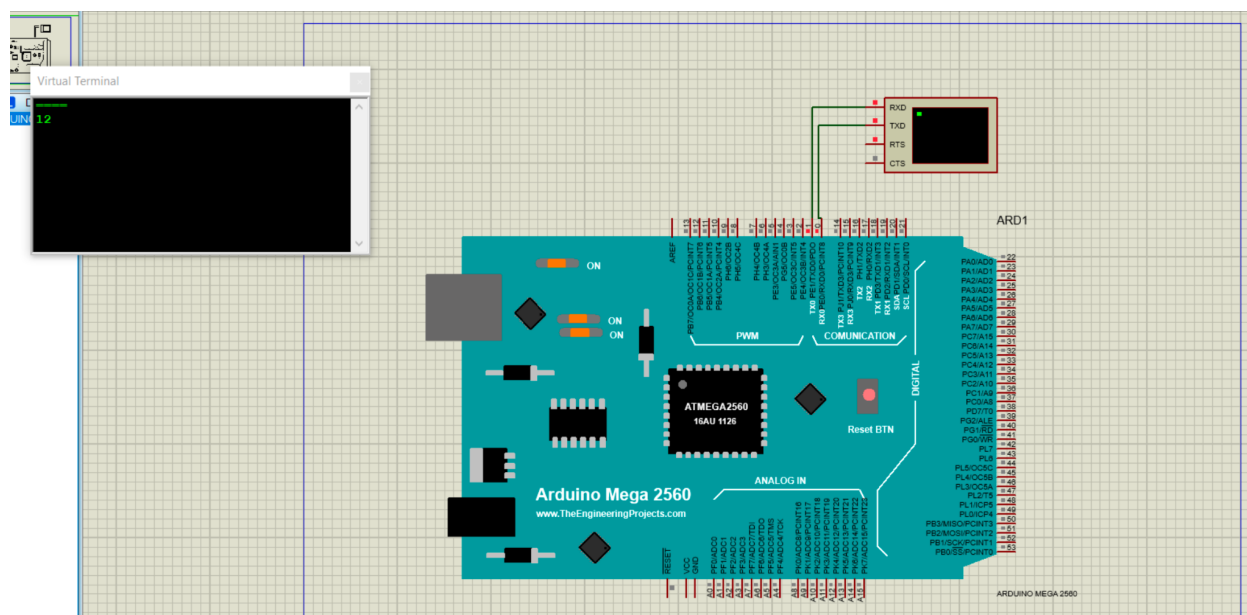
ا. کد کنترل کننده مدل فوق را در قالب یک تابع برای برد Arduino Mega2560 تولید کنید. تابع تولید شده را با استفاده از یک پروژه PlatformIO کامپایل کرده و با اجرای تابع step سطح بالا برای چندین نوبت و میانگین‌گیری، زمان اجرای تابع را در شبیه‌ساز Proteus برای سطوح مختلف بهینه‌سازی کامپایلر (00-، 01-، 02-، 03- و -Os) جداگانه اندازه‌گیری کنید. از پروژه خود تا به اینجا یک نسخه پشتیبان تهیه کنید.

در ابتدا، یک پروژه platformio می‌سازیم و کد جنریت شده را در main قرار می‌دهیم. (توجه شود که باید تابع setup را خودمان اضافه بکنیم)

تابع setup را به صورت دستی به کد اضافه می‌کنیم. لازم از تا تابع پله را ۱۰۰ بار اجرا کنیم البته همانطور که در صورت سوال نیز مطرح شده است باید تا با استفاده از تابع micros زمان اجرا را اندازه‌گیری بکنیم. البته این اندازه‌گیری کافی نیست و هدف از اندازه‌گیری به دست آوردن میانگین است. که این مقدار را حساب کرده و در خروجی نمایش می‌دهیم. در ادامه تابع پیاده‌سازی شده آمده است:

```
void setup() {  
    Serial.begin(9600);  
    untitled_initialize();  
    unsigned long sum = 0;  
    int count = 100;  
    for (int i = 1; i <= count; ++i){  
        unsigned long start = micros();  
        untitled_step();  
        unsigned long end = micros();  
        unsigned long delta = end - start;  
        sum += delta;  
    }  
    Serial.println((sum/count));  
}
```

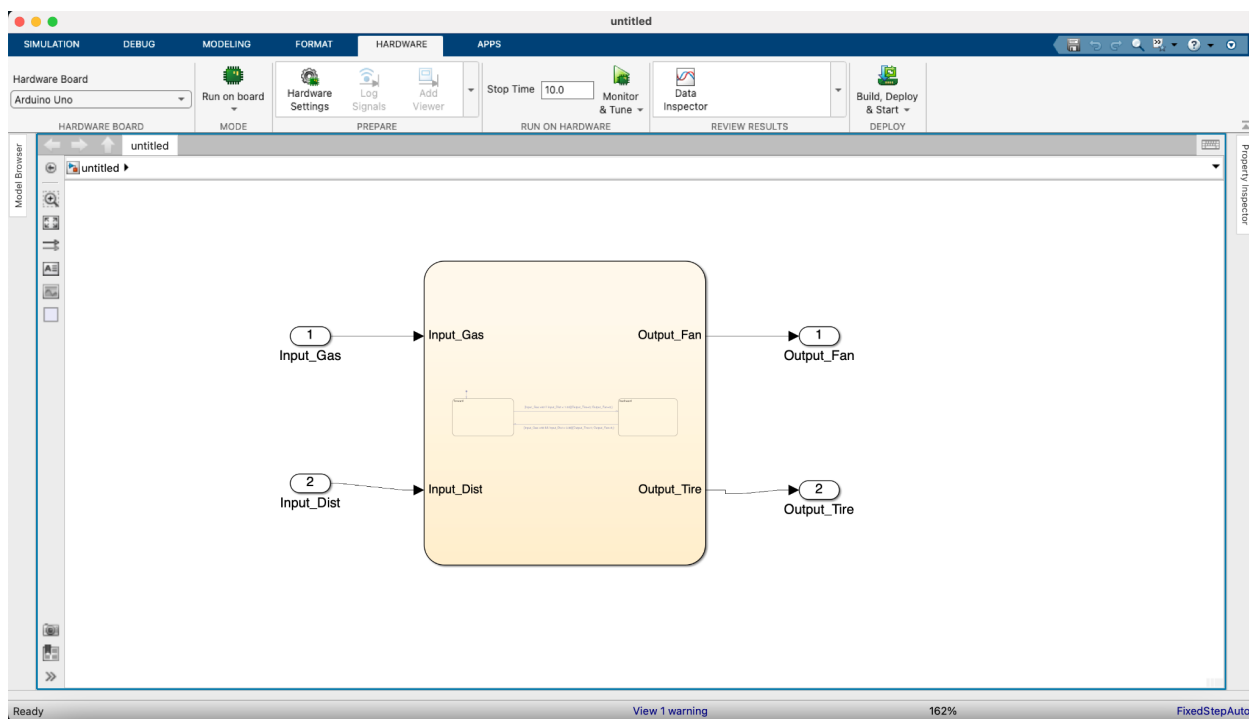
به ۶ روش خواسته شده کامپایل می‌کنیم و در تمام حالت‌ها خروجی زیر تولید شده است. (تفاوتی نداریم)



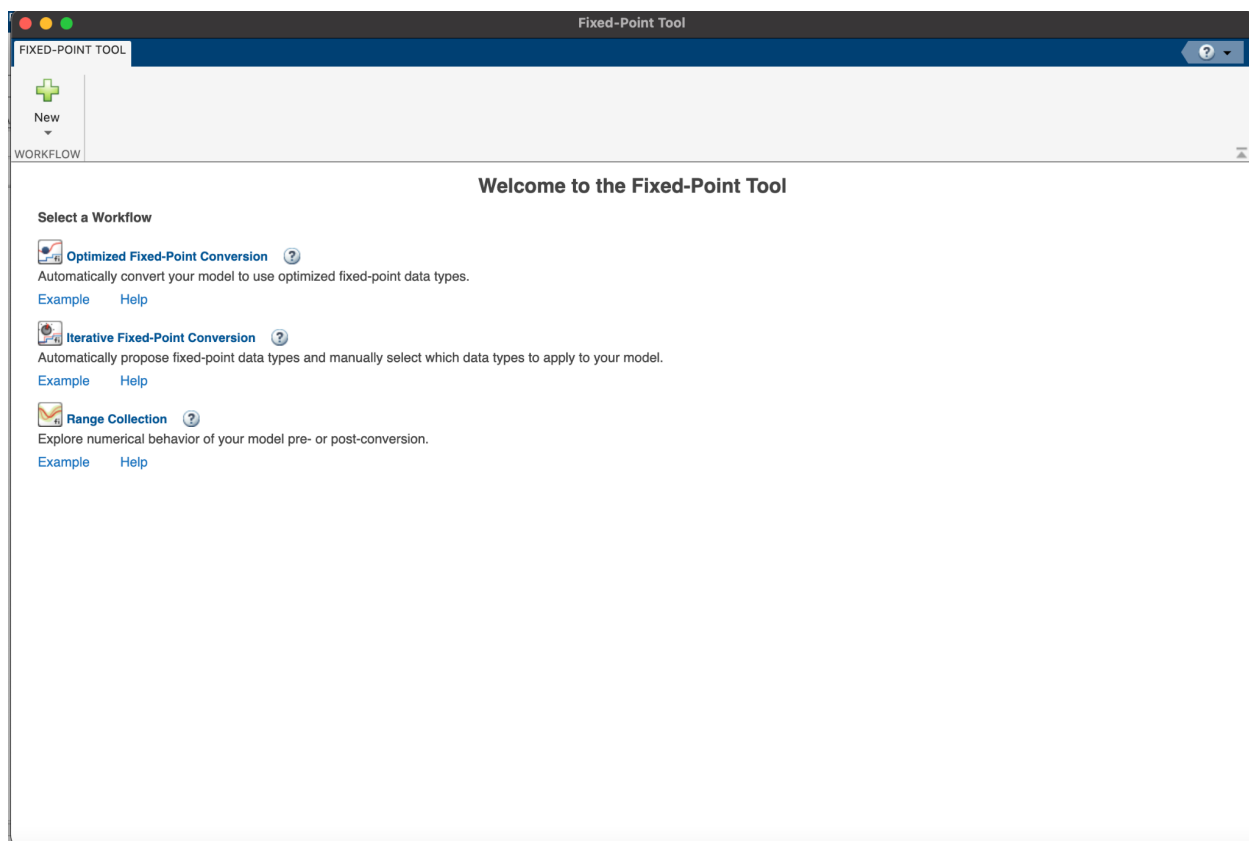
ب. با استفاده از ابزار ¹²Fixed-Point Designer، عملیات زیر را برای کنترل کننده انجام دهید. ممکن است

نیاز باشد برخی بلوک‌های خروجی را موقتاً از مدل خارج کنید.

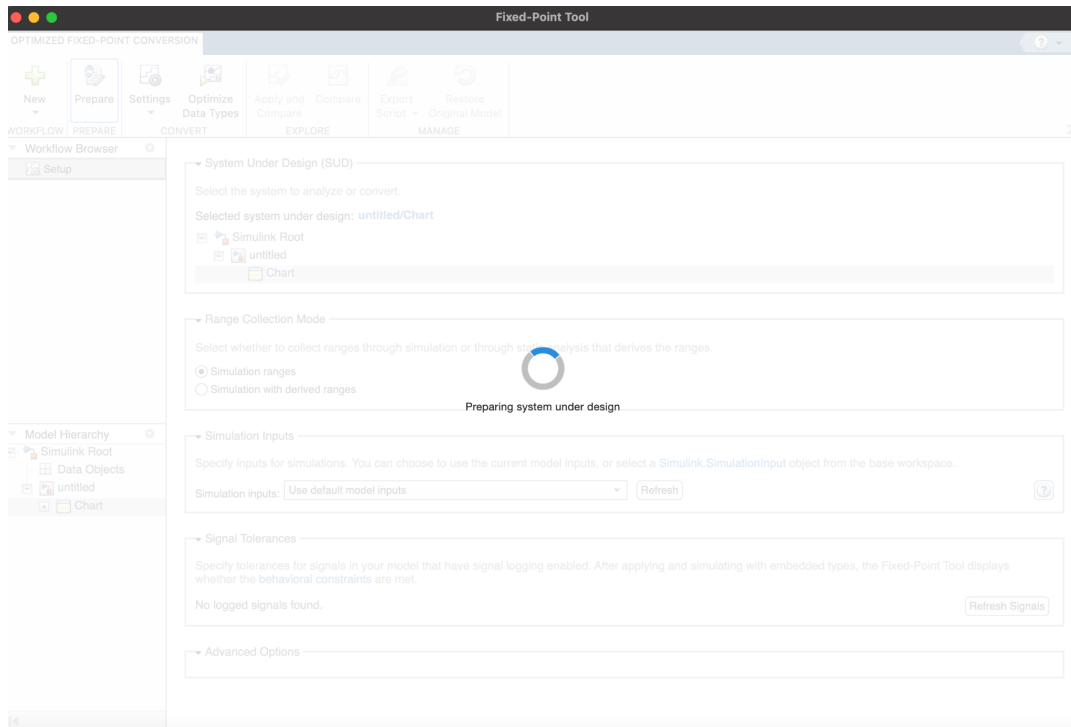
- آماده‌سازی مدل برای آنالیز (درج خودکار ورودی/خروجی‌ها برای تبدیل نوع)؛
- انجام شبیه‌سازی برای جمع‌آوری رنج و دقت سیگنال‌ها؛
- پیشنهاد داده‌های نوع ممیز ثابت به‌جای نوع‌های ممیز شناور؛ و
- اعمال تغییرات در مدل و جایگزینی نوع‌های جدید در مدل.



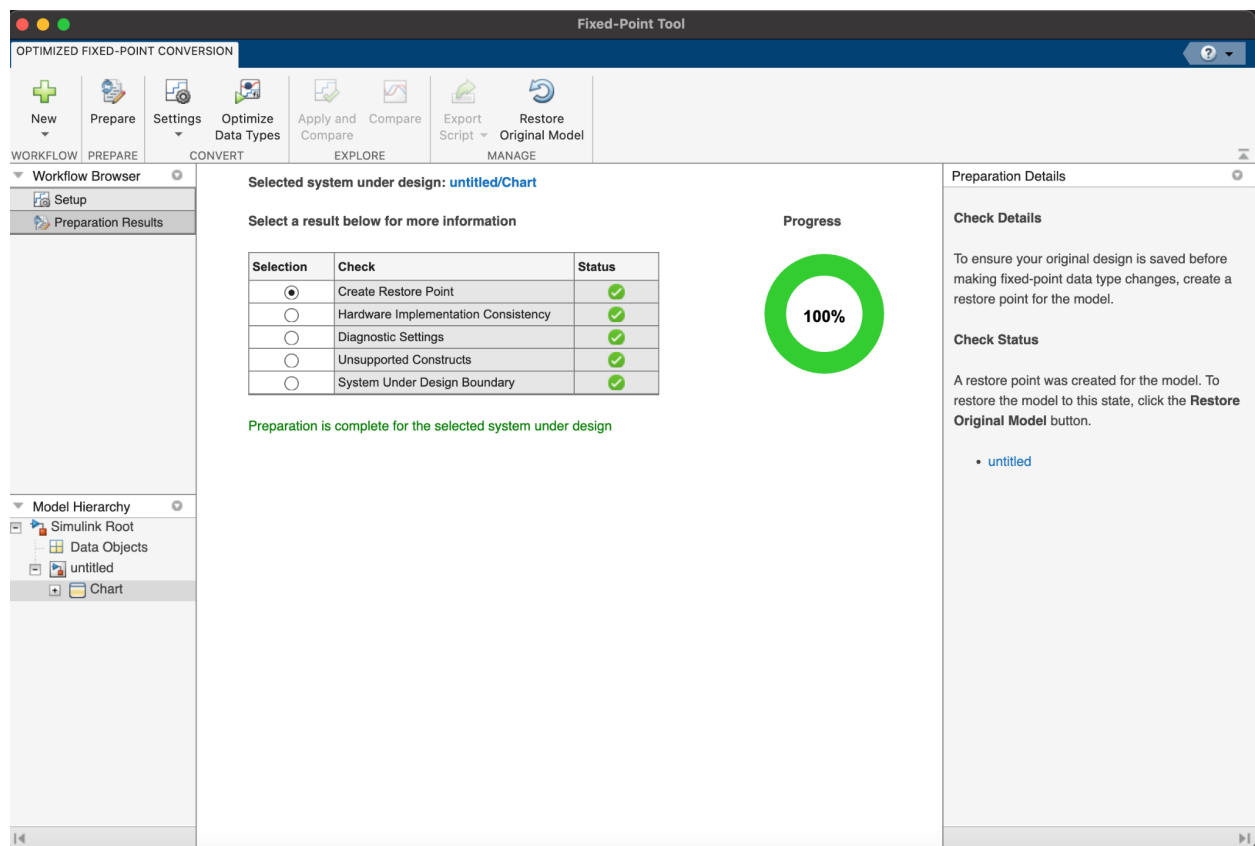
در این بخش سوال نیاز است تا از ابزاری به نام Fixed Point Designer استفاده کنیم. در این مرحله به صورت تصویری تمام مراحل مورد نیاز را نمایش داده ایم و دست آخر با استفاده از ابزاری به نام Coder کد جدید ایجاد شده را Build کرده ایم.



در مرحله اول بر روی Subsystem ای که قصد داریم بر روی آن ابزار Fixed Point Designer را اجرا کنیم انتخاب کرده و سپس با کلیک راست و انتخاب ابزار مورد نظر به صفحه فوق میرویم. طبق تصویر بالا روند مورد نظر خود را انتخاب کرده و ادامه می دهیم. در مرحله بعدی باید مدل مورد نظر خود را آماده سازی کنیم (که با زدن دکمه Prepare) این اتفاق می افتاد.



مدتی برای آماده سازی باید صبر کنیم تا در نهایت به مرحله زیر برسیم:



همانطور که مشخص است مدل کاملاً آماده است. تا به اینجای کار مرحله اول ذکر شده در صورت سوال انجام شده است.

Fixed-Point Tool

ITERATIVE FIXED-POINT CONVERSION

Workflow: WORKFLOW | PREPARE | **COLLECT** | CONVERT | VERIFY | MANAGE

Tools: New | Prepare | **Collect Ranges** | MATLAB Functions | Propose Data Types | Apply Data Types | Simulate with Embedded Types | Run to compare in SDI | Compare Results | Restore Original Model

Collect Ranges
Collect simulation ranges using data type override

Collect ranges using:

- ☒ **Use current settings**
Use current data type override set on the model
- ☐ **Double precision**
Override data types with doubles
- ☐ **Single precision**
Override data types with singles
- ☐ **Scaled double precision**
Override data types with scaled doubles

Preparation is complete for the selected system under design

Progress
100%

Check Details
To ensure your original design is saved before making fixed-point data type changes, create a restore point for the model.

Check Status
A restore point has previously been created for this model. To restore the model to this state, click the **Restore Original Model** button.

Model Hierarchy
Simulink Root
Data Objects
untitled
Chart

Fixed-Point Tool

ITERATIVE FIXED-POINT CONVERSION

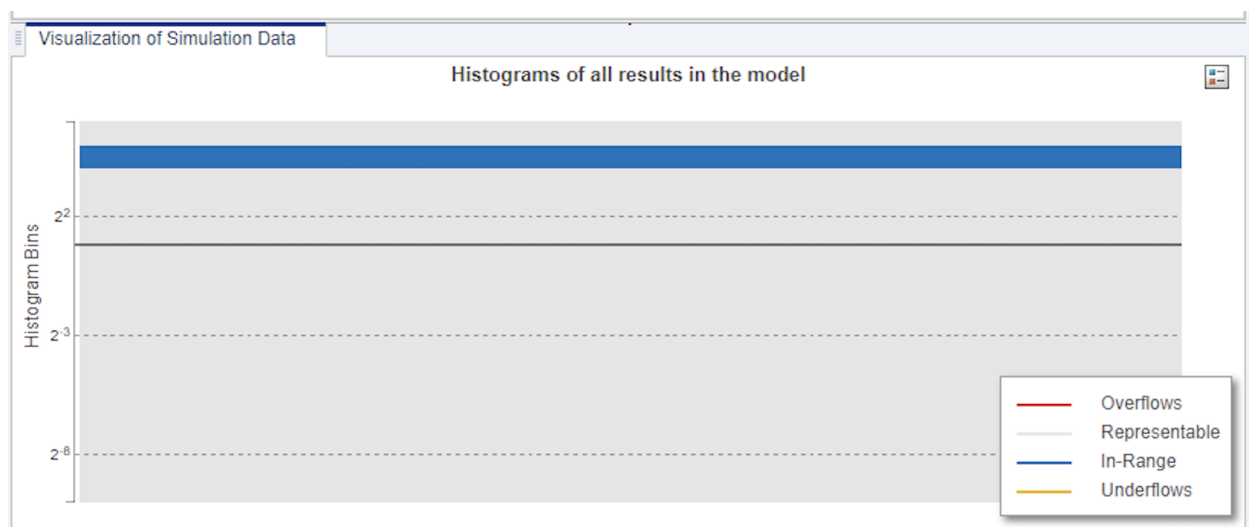
Workflow: WORKFLOW | PREPARE | COLLECT | **CONVERT** | VERIFY | MANAGE

Tools: New | Prepare | Collect Ranges | MATLAB Functions | Propose Data Types | Apply Data Types | **Simulate with Embedded Types** | Run to compare in SDI | Compare Results | Restore Original Model

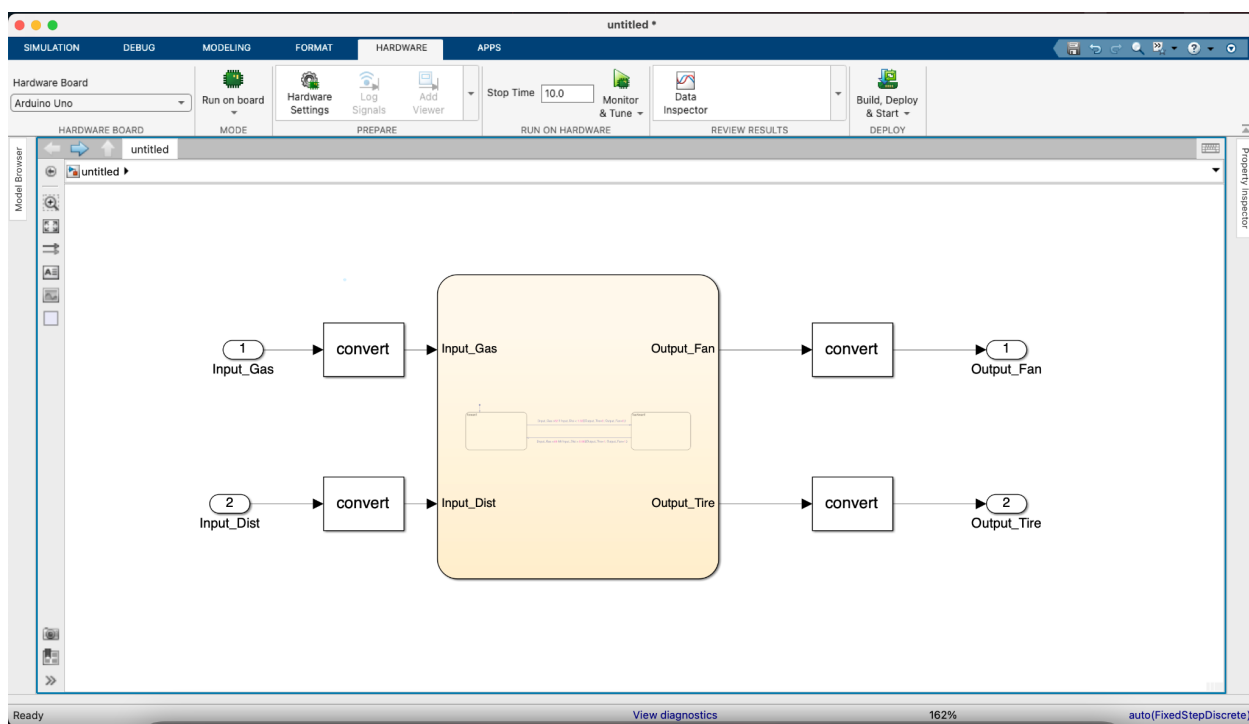
Results

Name	CompiledDT	SpecifiedDT	ProposedDT	Accept	SimMin	SimMax
Chart1.degree_output	fixdt(1,16,4)	fixdt(1,16,4)			0	0
Chart1.force_input	fixdt(1,16,4)	fixdt(1,16,4)			0	0
Chart1.frequency_output	fixdt(1,16,10)	fixdt(1,16,10)			-20	0
Chart1.temp_input	fixdt(1,16,4)	fixdt(1,16,4)			0	0
Chart1.timerforce	fixdt(0,16,14)	fixdt(0,16,14)			0	2
Chart1.timertemp	fixdt(0,16,15)	fixdt(0,16,15)			0	1
Data Type Conversion	fixdt(1,16,4)	fixdt(1,16,4)			0	0
Data Type Conversion1	fixdt(1,16,4)	fixdt(1,16,4)			0	0
Data Type Conversion2	fixdt(1,16,4)	fixdt(1,16,4)			0	0
Data Type Conversion3	fixdt(1,16,10)	fixdt(1,16,10)			-20	0

Model Hierarchy
Simulink Root
Data Objects
untitled
Chart

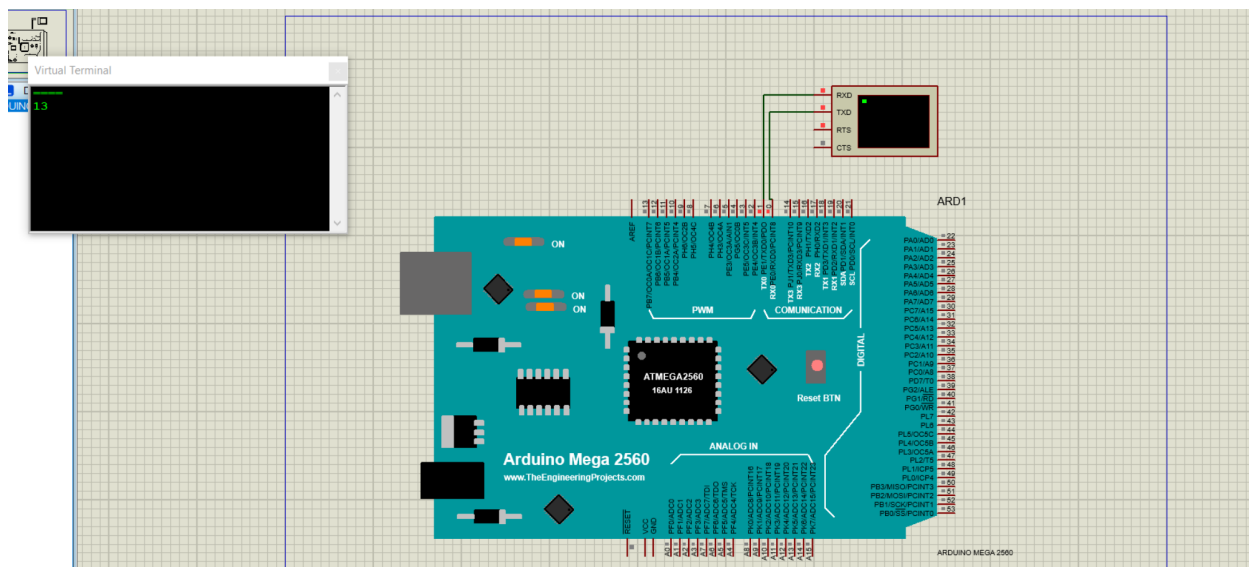


تمام مراحل خواسته شده در موارد بالا انجام شده است و در انتها نوع های جدید در مدل جاگذاری شده است.
تصویر زیر نشان میدهد که از Convert ها استفاده شده است در انتها.



ج. تولید کد را مجدداً برای مدل ممیز ثابت انجام داده و مطابق بخش آن را برای برد کامپایل و شبیه سازی کنید. زمان اجرا را در این حالت نیز برای سطوح مختلف بهینه سازی کامپایلر جداگانه اندازه گیری کنید. زمان های اجرای اندازه گیری شده را در قالب جدولی در گزارش خود تنظیم کنید.

برای این قسمت لازم است تا صرفاً تمام مراحل قسمت الف را دوباره تکرار بکنیم و مجدداً برای خلاصه بودن گزارش در تمام ۶ حالت کامپایل شده خروجی همانند زیر داریم.



نتیجه گیری به این صورت است که بعد از شبیه سازی برای بار دوم در بهینه سازی های کامپایلر تاثیری نداشته است. و صرفاً زمان اجرا ۱ واحد افزایش داشته است (همانطور که در شکل بالا مشخص است)