کتابچه راهنمای نرم افزار فنی– مهندسی

DIgSILENT *PowerFactory*
Version 13.0

شرکت برق منطقه ای فارس

معاونت برنامه ریزی و تحقیقات

۱۳۸۴

تهیه شده در :
شیراز-خیابان زند-نبش خیابان فلسطین
شرکت برق منطقه ای فارس
تلفن : ۹-۲۳۳۰۰۳۱ (۰۷۱۱)
فاکس : ۲۳۵۹۰۴۷ (۰۷۱۱)
www.frec.co.ir

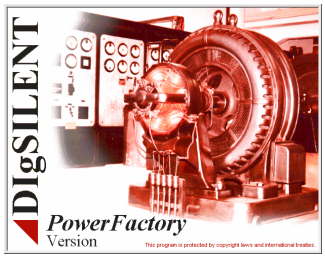
تماس با مترجمان
محمدرضا گلساز شیرازی mshirazi@frec.co.ir
احمد فرشچیان فسایی farshchian@frec.co.ir

Advanced User's Manual

فهرست مطالب

Chapter 3

Programming Primitive Block Definitions



مدل سازی شبکه به منظور آنالیز پایداری یکی از اصلی ترین مسائل در زمینه آنالیز شبکه قدرت می‌باشد. بسته به دقت مدل اجرا شده، اعتبار سیگنالهای فشارقوی با دامنه بزرگ، در دسترس بودن پارامترهای شبکه و آزمایشات یا خطاهای اعمال شده، تقریباً می‌توان هر نتیجه ای را ایجاد کرد و بنابراین باید پارامترهای (آرگومان های) مورد نیاز برای تحقق آنها را یافت.

یک مثال ساده می‌تواند این‌حقیقت را روشن کند. یک شبکه قدرت ۱۰ گیگاوات را در نظر بگیرید، سؤالی که مطرح است در مورد انحراف فرکانس حالت کار دائم سیستم است زمانیکه کل بار یک واحد ۲۰۰۰ مگاواتی را از دست داده باشیم، که این بستگی زیادی به ضریب وابستگی فرکانس بارها K_f دارد. فرض نمایید کل افت شبکه ۷ درصد و K_f برابر صفر باشد، آنگاه انحراف فرکانس حالت ماندگار تقریباً ۷۰۰ میلی هرتز خواهد شد. حال با در نظر گرفتن یک ضریب واقعی تر $K_f = 5\%/_{Hz}$ ، انحراف فرکانس حالت ماندگار مورد انتظار تنها ۵۹۶ میلی هرتز می باشد. از طرف دیگر، وابستگی فرکانس ممکن است کمی بالاتر یا پایین تر باشد، اما مشخصات غیرخطی راندمان توربین آبی و رفتار غیرخطی سوپاپ بخار می‌تواند در یک نقطه کار معین از ظرفیت بارگیری واحد، وابستگی شدیدتری به یکدیگر داشته باشند. درنتیجه مادامی که فقط یک یا دو سناریوی بارگیری مختلف در نظر گرفته شود، مقادیر متوسط با مدلهای ساده منطقی ممکن است نتایج قابل قبولی را ایجاد نمایند که بوسیله تنظیم کردن فقط تعدادی پارامتر کلیدی مثل ضرایب وابستگی فرکانسی بارها یا تنظیمات افت بدست آمده اند.

این مسئله را بخاطر داشته باشید که ساختار های مدل شبکه وتنظیمات پارامتر باید با توجه به معیارهای اصلی زیر محاسبه شوند :

System size (اندازه شبکه) : شبکه های کوچک و بزرگ دارای پارامترهای کلیدی مختلفی هستند. به مثال بالا رجوع کنید، برای یک شبکه کوچکتر قدرت وابستگی فرکانس بارها به طور کلی مطرح نمی‌باشد در حالی که در شبکه های بزرگ مثل *UCPTE* یا *UPS* وابستگی فرکانس ممکن است به اندازه کل نیاز ظرفیت ذخیره چرخشی شبکه تحت پوشش باشد.

Unit size (اندازه واحد) : رفتار حالت گذرا و حالت ماندگار واحدهای بزرگ در تعیین پاسخ کل شبکه بسیار نقش تأثیرگذارتری نسبت به واحدهای کوچکتر که عملاً تأثیری بر کل شبکه نمی‌گذارند، دارند.

System structure (ساختار شبکه) : مستقل از شبکه و اندازه واحد، ساختار شبکه پیش از هر عامل دیگری مطرح می‌باشد. این مسئله را وقتی که شبکه های ضعیف گسترده در طول جغرافیایی یا دارای زیرساختهای اختصاصی آنالیز می‌شوند، می‌توان به راحتی درک کرد.

System fault (اختلال شبکه) : عمده ترین مسائل مطرح در مدل سازی شبکه، خطاهای اعمال شده و مسائل مربوطه است که باید مورد نقد و بررسی قرار گیرند. آنالیز میرایی شبکه وتنظیم pss لزوماً نیازی به در نظر گرفتن دینامیک بویلر ندارد. از طرف دیگر اگر منحنی های دراز مدت یا میان مدت

عناصر وابسته شبکه حذف شده باشند و یا چشم پوشی گردند، بهینه سازی مانور قطع بار و بازیابی فرکانس، نتایج مناسبی را ایجاد نخواهند کرد.

Study purpose (هدف مطالعه) : بطور کلی برای شبکه هایی که در مرحله طراحی هستند، مادامیکه هیچ اطلاعات خاص اضافه دیگری موجود نباشد می توان از مدلها وپارامترهای نمونه استفاده نمود. به هرحال در موارد توسعه و گسترش شبکه لازم است تا از مدلهای دقیق تر استفاده گردد. یعنی جایی که مدل دقیق تر باید بخشی از مشخصات شبکه را تشکیل دهد، به آنالیز مسائل بهره برداری و بهینه‌سازی عملکرد باید توجه خاصی مبذول گردد. در این موارد، مدل سازی دقیق مؤلفه های مربوطه ضروری و لازم الاجرا می‌باشد.

به محض این که یک آنالیز و ارائه دقیق مدلهای سیستم لازم باشد، اولین سوالاتی که بلافاصله مطرح می‌شوند، عبارتند از :

- چگونه می‌توان ساختارها و پارامترهای مدل را تعیین کرد؟
- آیا مدلهای *IEEE* و سایر بلوک دیاگرام های سازندگان تجهیزات شبکه، دقیق و مناسب هستند؟
- چطور می‌توان اطلاعات موجود را در داخل نرم افزار آنالیز شبکه قدرت استفاده کرد؟

روشی که در این جا ارائه شده و به صورت موفقیت آمیزی در پروژه های مختلف بکار رفته است را می‌توان *'Advanced System Modeling Approach (ASMA)'* نامید. کابردهای نمونه آن عبارتند از :

- آنالیز مشکلات کنترل کننده و عملکردهای غلط آن، مخصوصاً تحت شرایط اغتشاش
- بهینه سازی تنظیمات پارامتر کنترل
- مدل سازی ساختارهای غیر عادی شبکه و مفاهیم کنترل که اغلب در سیستم های صنعتی یافت می‌شوند.
- انجام مطالعات فاز صفر طراحی و ارائه مشخصات اجزاء و سیستم ها (از قبیل *PSS*، کنترل‌کننده ولتاژ و *HVDC*)

برای روش *ASMA* مراحل زیر دارای اهمیت بنیادی می‌باشند :

setup of system models (نصب مدلهای شبکه) : بر اساس معادلات اساسی مهندسی و فیزیکی، معادلات جبری و دیفرانسیل پایه، باید در محدوده دقت مورد احتیاج تنظیم شوند. بعلاوه همه پارامترها مانند ثابت زمانی و ضرایب بهره که می‌توانند از این معادلات پایه بدست آیند، نیز باید به همان شیوه محاسبه گردند.

performance of system tests (تست های اجرای سیستم) : به منظور تعریف همه پارامترهای دیگر بخصوص منحنی های غیرخطی، تست های اجرای شبکه، بهترین روش می‌باشند. در اکثر موارد، آزمایشات پاسخ فرکانسی اجازه تعیین هر ساختار غیرخطی و پارامترهای آن را نمی‌دهند. روشهای تست ویژه، که با عملکرد عادی تداخلی نداشته باشند باید به منظور تمرکز بر روی مشخصات حالت ماندگار، ضرایب بهره و ثابت زمانی ها بکار برده شوند. این اندازه گیریها ترجیحاً توسط یک سیستم اندازه گیری دیجیتالی حالت گذرای بسیار دقیق انجام می شوند.

system identification (شناسایی شبکه) : روشهای شناسایی غیر خطی، چند ورودی و چند خروجی برای روشهای شناسایی شبکه بکار برده می‌شوند. برای نمونه، عدم تطابق بین منحنی های اندازه گیری شده وشناسایی شده باید کمتر از ۲ ٪ باشد.

Comparison of measurments and simulations (مقایسه بین اندازه گیری ها و شبیه‌سازی ها) : علاوه بر آنالیز زیر شبکه ها ومؤلفه ها، عملکرد کل شبکه نیز باید با مدل تئوری و برای همه حالتهای بهره برداری مربوطه مقایسه شود.

البته کاربرد روش 'ASMA' به شکل کاملاً جدی و سختگیرانه برای مدلسازی رله ها و برای پیچیدگی های کمتر یا توابع کنترل دیجیتالی لازم نمی‌باشد، زیرا اینها به صورت واضح مطابق با اسناد آزمایشات تأیید شده و کلی مربوط به خودشان تعریف شده اند. اما مستقل از شبکه آ نالیز شده، در جایی که نمایش شبکه نمی‌تواند با مدلهای کلاسیک *IEEE* یا هر مدل استاندارد دیگر تطابق داشته باشد، نیاز اساسی به استفاده از یک روش آسان و قابل انعطاف برای درک تمام مدلهای اختصاصی وجود دارد.

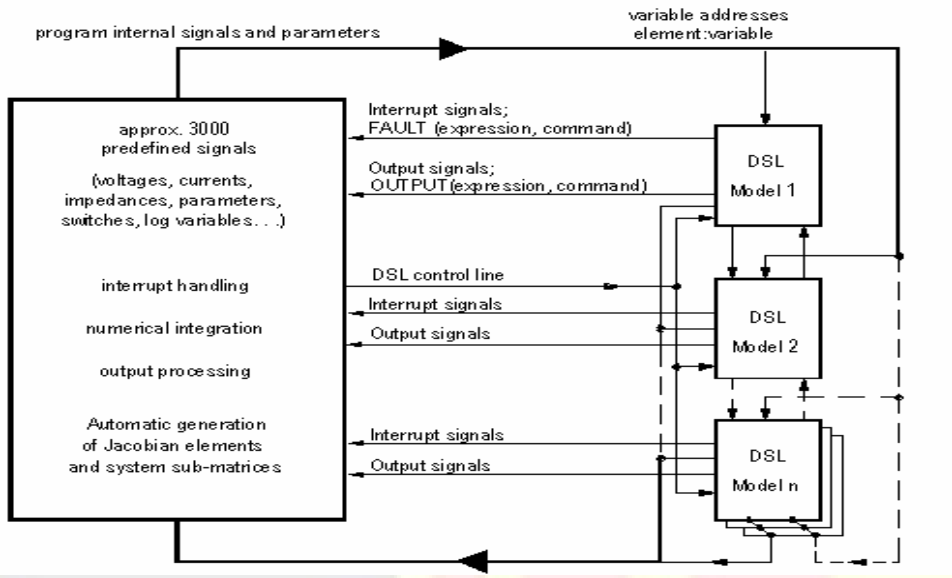
3.1 Modelling and Simulation Tools

همانطور که قبلاً بیان شد، ضروری ترین و تعیین کننده ترین عامل برای دستیابی به نتایج شبیه سازی مطمئن، دقت و تکامل مدل های ارائه شده برای شناسایی شبکه و اهداف شبیه سازی می‌باشد. روشهای انجام این خواسته متغیر بوده و از روشهای کلاسیک و سنتی استفاده از نرم افزارها که به عنوان محیط واسطه برای مدلهای تعریف شده توسط کاربر عمل می کردند و در سطح نرم افزارهای فرترن و C "معمولاً به شکل فهرست های اتصال" بودند، تا استفاده از روش های بلوکی که بر اساس تهیه بلوک ماکروهای سطح پایین از پیش تعریف شده عمل کرده و به سطح تعریف مورد متصل می باشند، گسترده می باشند. بعلاوه اغلب ابزارهای شبیه سازی مدرن تجارتی همه منظوره موجود، ممکن است برای نمایش ویژه و انعطاف پذیر شبکه استفاده گردند. متأسفانه روش منحنی های پخش بار، شبکه های الکتریکی خاص را به حد کافی پوشش نمی‌دهد.

به منظور ایجاد یک ابزار شبیه سازی و مدل سازی قابل انعطاف، که قسمتی از برنامه پایداری را تشکیل می دهد، یک سیستم کنترل مبتنی بر زبان شبیه سازی ایجاد شده است. که ویژگیهای اصلی زیر از زبان شبیه سازی *DIgSILENT* در آن منظور شده اند :

- ابزار شبیه سازی در طبقه یک زبان شبیه سازی پیوسته شبکه (*CSSL*) قرار داده می‌شود.
- *DSL* شامل یک توصیف ریاضی کامل از شبکه‌های خطی و غیرخطی پیوسته (زمانی) می‌باشد.
- ابزار شبیه سازی که برپایه کنترل مشترک و نمودارهای منطقی بنا شده است به یک زبان بدون رویه سوق داده می‌شود که درآن توالی عناصر را می‌توان به دلخواه انتخاب کرد. به عبارت دیگر، یک مدل *DSL* می‌تواند به یک ارائه گرافیکی تبدیل شود.
- تهیه و ایجاد یک تعریف انعطاف پذیر از ماکروها که می‌توانند : معادلات جبری، عناصر اصلی کنترل مانند *PID* یا *PTn* یا حتی زیر سیستم های فیزیکی کامل مثل انواع گروه های سوپاپ یا سیستم های تحریک کننده را در برگیرد.

- تهیه توابع درونی مختلف مثل : "*lapprox*" , "*limits*" , "*lim*" , "*select*" "*picdrop*" به منظور فراهم کردن یک کنترل کامل بر مدلها.
- ایجاد روش های رسمی مختلف برای شناسایی خطا و اهداف آزمایشی مثل : شناسایی حلقه جبری، گزارش متغیرهای بکار نرفته وتعریف نشده و شرایط اولیه حذف شده.



شکل ۳-۱ : ساختار سیستم Digilent DSL

3.2 DSL Implementation: an Introduction

زبان شبیه سازی *DIgSILENT* به منظور تعریف کنترل کننده های دینامیکی جدید که سیگنالهای ورودی را از شبکه قدرت شبیه سازی شده دریافت کرده و با تغییر برخی سیگنال های دیگر عکس العمل نشان می‌دهد، استفاده می‌شود.

DSL خودش می‌تواند به عنوان یک ابزار اضافه شونده به تابع آنالیز گذرای *DIgSILENT* اضافه شود. در زمان شبیه سازی، معادلات نمونه مدلهای *DSL* با آنهایی که رفتار دینامیکی مؤلفه های شبکه قدرت را توصیف می‌کنند، ترکیب می‌شوند. سپس این معادلات بصورت توأمان حل گردیده و به یک شبیه ساز گذرای جامع و یکپارچه مجموعه شبکه قدرت و کنترلرهای آن تبدیل می‌شوند.

توابع واسطه اصلی *DSL* عبارتند از :

Signal input and output channels (کانلهای ورودی و خروجی سیگنال) : هر متغیر تعریف شده در برنامه اصلی (در واقع بیش از ۲۵۰۰) و در مدل *DSL*، می‌تواند قابلیت خواندن و نوشتن داشته باشد. ویژگیهای آدرس اصلی وفرعی برای امکان دسترسی به هر سیگنال موجود در سیستم یا جمع کردن ساختارهای پیچیده مثل ماژول های سخت افزاری تجهیزات دربرگیرنده تجهیزات *rack* (قفسه) و *Function card* (کارت تابع)، ذکر شده است.

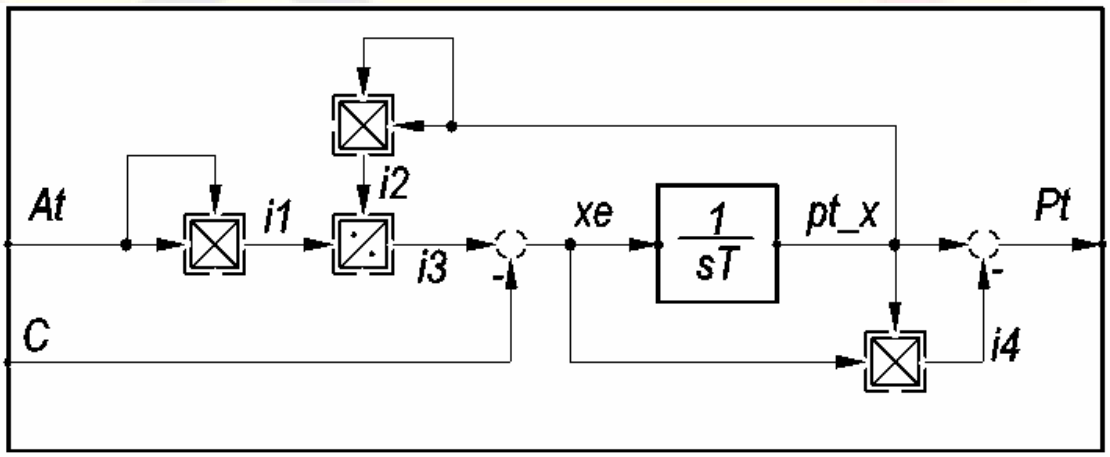
Events (رخدادها) : شرایط ارزیابی شده بوسیله مدلهای *DSL* ممکن است باعث ارسال رخدادها به برنامه مرکزی شوند جایی که آنها باید به نوبت ارزیابی گردند.

Output and Monitoring (خروجی و نظارت) : شرایطی که منجر به فعال سازی نمایش پیغامهای تعریف شده توسط کاربر در پنجره خروجی می شود.

ساختار مدل *DSL* از طریق مثال به خوبی توصیف گردیده است. مثال استفاده شده مربوط به مدل واحد محرک اولیه یک توربین آبی ساده می‌باشد. این مدل *DSL* به صورت گرافیکی تعریف شده است، و دارای یک ماکروی جاگذاری شده *DSL* می‌باشد. آن ماکروی جاگذاری شده یک انتگرالگیر ساده را مدل می کند و با برنامه نویسی آن تعریف شده است. روش اصلی طراحی مدلهای جدید *DSL* عبارت است از :

۱- یک مجموعه از مدلهای *DSL* پایه ایجاد شده است. این مدلها شامل کنترل کننده های ابتدایی ساده مثل یک کنترل کننده تأخیری مرتبه اول (*First order time lag*) یا یک کنترل کننده *PID* می باشند. برنامه *DIgSILENT* به همراه تعداد زیادی از مدلهای کنترل کننده های ابتدایی برای کاربران ارسال می‌شود. کنترل کننده های ابتدایی جدید بوسیله برنامه نویسی معادلات دیفرانسیل آنها و تنظیمات سیگنال با استفاده از زبان *DSL* ایجاد می‌شوند.

۲- کنترل کننده های پیچیده تر به صورت گرافیکی بوسیله ترسیم بلوک دیاگرام آن ایجاد می‌شوند. چنین بلوک دیاگرامی معمولاً از ارجاعاتی برای مدلهای دیگر *DSL* استفاده می‌کند، که بنابراین یک کنترل‌کننده پیچیده تر را ایجاد می نمایند. ارجاعات کنترل کننده را می‌توان برای گنجاندن مدلهای ابتدایی *DSL* در مدلهای پیچیده نیز استفاده کرد، اما همچنین ممکن است به دیگر مدلهای پیچیده تعریف شده گرافیکی هم ارجاع داده شوند. بنابراین ممکن است کنترل کننده های بسیار پیچیده در یک روش سلسله مراتبی بوسیله طراحی مدلهای فرعی و زیرمجموعه های فرعی تر آنها تا جایی که پایین ترین سطح آن را کنترل‌کننده های ابتدایی *DSL* تشکیل می دهد، طراحی شوند.



شکل ۳-۲ : نمودار مدل ساده یک توربین آبی

شکل ۳-۲، نموداری را نشان می دهد که برای تعریف مدل توربین آبی استفاده شده است. وقتی یک *graphics rebuild* اجرا می‌شود، نظیر آنچه در پنجره خروجی نشان داده شده است، کد *DSL* منتجه بوجود خواهد آمد :

```
1. model pt = 'pmu - hydro ' (At , C; x1;Ti;)
```

```
2.      pt - x  = 'I.  BlkDef ' (xe ; x1 ;Ti;)
3.      i3 = i1 / i2
4.      i1 = At * At
5.      i2 = pt - x*pt - x
6.      i4 = xe * pt - x
7.      xe = i3 - c
8.      pt = pt - x - i4
```

شماره های خطوط برای سهولت عمل اضافه شده اند. تعریف بلوک متناظر نشان می‌دهد که :

Output signals : pt سیگنال خروجی

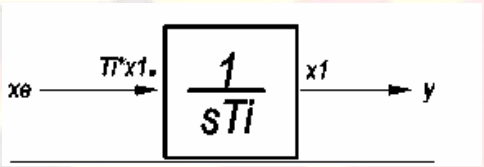
input signal : At ,C سیگنال ورودی

State variables : X1 متغیرهای وضعیت

Parameter : Ti پارامتر

Internal variables : متغیرهای داخلی

این مثال، مدل یک توربین آبی ساده را با سیگنال های ورودی *At* و *C* و سیگنال خروجی *Pt* توصیف می‌کند.



شکل ۳-۳ : نمایش گرافیکی یک مدل *DSL* یک انتگرالگیر

درشکل ۳-۳، نمایش گرافیکی مدل *DSL* ابتدایی جاگذاری شده ترسیم شده است. این مدل ابتدایی در توربین آبی بکار رفته است (در سطر ۲ از تعریف توربین). این *DSL* ابتدایی از یک انتگرالگیر تکی استفاده نموده است و به شکل زیر برنامه نویسی شده است :

```
1.  model y = 'I'  (xe ; x1 ; Ti;)
2.          [Ti] = 's'
3.   limits (Ti) = (0,)
4.   inc (x1) = y
5.  inc (xe) = 0
6.   x1. = xe/ Ti
7.   y = x1
```

خط ۱ با فشار دادن دکمه ***Equations*** از کادر محاوره ای ویرایش بلوک دیاگرام ایجاد شده است.

خط ۲ تا ۷ به صورت دستی وارد شده اند.

کادر محاوره ای تعریف بلوک برای تنظیم موارد زیر استفاده شده است :

Output signals : y

Input signals : xe

State variables : x1

Parameter : Ti

Internal Variables :

۳-۲-۱ قسمتهایی از یک مدل DSL

هر دو مثال *DSL* دو قسمت اصلی هر مدل *DSL*، ابتدایی یا پیچیده را نشان می‌دهند :

۱. تعاریف واسطه

۲. توصیف مدل *DSL*

Interface description (توصیف محیط واسطه) : واسطه نام مدل، نام سیگنال های ورودی و خروجی، پارامترهای مدل و متغیرهای حالت را تعریف می‌نماید. این موارد در پنجره خروجی در عنوان مدل نشان داده شده اند.

مثال (سطر ۱ از مدل توربین آبی) :

model pt = pmu - hydro (At , C;x1 ; Ti ;)

کادر محاوره ای بلوک دیاگرام افزون براین اجازه تعریف پارامترهای محدود سازی و سیگنال های ورودی، و طبقه‌بندی مدل به عنوان یک مدل خطی و یا یک ماکروی *DSL* را امکان پذیر نموده است.

Model description (توصیف مدل) : توصیف مدل، مدل *DSL* را بر اساس سیگنال های تعریف شده در واسطه وصف کرده و تشریح می‌نماید. توصیف *DSL* شامل موارد زیر می‌باشد :

- توصیفات پارامتر : نام و واحد
- دامنه های مجاز پارامتر
- شرایط اولیه و توابعی که برای محاسبه مقادیر اولیه استفاده می‌شوند.
- روابط جبری که کنترل کننده را تعریف می‌کنند.

مثال (انتگرالگیر) :

```
2.        [ Ti] = ' s '  ! the unit of Ti is seconds
3.  limits (Ti) =  ( 0 )! Ti > 0
4.  inc (x1)  = y        ! initially x1 = y
5.  inc (xe) = 0         ! initially xe = 0
6.  x1.    = xe / Ti     ! equation 1 : &*1/ &t = xe / Ti
7.  y      = x1          ! equation 2 : y = x1
```

۳-۲-۲ ویژگیهای پیشرفته

انتگرال گیری عددی مدلهای 'DSL'، زمان بندی قطع و برنامه ریزی سیگنال ورودی- خروجی بطورخودکار بوسیله برنامه مرکزی اداره می‌شود. به علاوه اگرخروجی یک مدل 'DSL' یک جریان الکتریکی است که به جریان باس کلی مناسب اضافه می‌شود در این حالت اگر یک بار یا ژنراتور ایجاد شود، همه عناصر لازم جاکوبین برای روش همانند سازی تکرار بطورخودکار محاسبه می‌شوند.

ویژگی مفید دیگر 'DSL' الگوریتم انجام شده برای تنظیم عددی ماتریس شبکه برای اهداف محاسبه مقدار 'eigen' می‌باشد. در نتیجه وقتی مقادیر 'eigen' شبکه محاسبه شده یا زمانی که روش کاهش شبکه مدل به کار می‌رود(MRT) هر مدل انجام شده در سطح 'DSL' بطورخودکار درنظرگرفته می‌شود. البته هر تابع محدود سازی سیگنال بطورخودکار برای این روش محاسبه غیر فعال می‌شود. به علاوه

ورودیها و خروجیهای پارامترهای مدل، تشکیلات آن را از طریق منوهای پنجره ها و غیره هم بطورخودکار از این مدل 'DSL' گرفته می‌شود.

3.3 Defining DSL Models

یک مدل *DSL* جدید را هم می‌توان به وسیله وارد کردن کد *DSL* در قسمت معادله عنصر *Block definition* یعنی *BlkDef* ساخت و هم می توان یک بلوک دیاگرام گرافیکی جدید ایجاد کرد. هر دو روش یک عنصر تعریف بلوک را ایجاد خواهند نمود که در برگیرنده تعریف مدل *DSL* است.

بنابراین عناصر تعریف بلوک در فرآیند ساختن یک مدل *DSL* دو هدف زیر را دنبال می‌کنند :

- آنها لوازم و مقدماتی را ایجاد می نمایند که در آن می توان یک *DSL primitvie* جدید را تعریف نمود.
- آنها تعاریف و بخش هایی از بلوک دیاگرام مرکب ساخته شده به صورت گرافیکی و گرافیک نموداری را که برای تعریف مدل استفاده شده است، نگهداری می‌کنند.

DSL ابتدایی و اصلی، بلوکهایی هستند که از آنها نمودارهای کنترل کننده های بسیار پیچیده ساخته می‌شود. برای مثال، یک *DSL* ابتدایی، ممکن است از یک فیلتر پایین گذر استفاده کرده باشد، که می‌تواند بعداً برای ساخت کنترل کننده های بسیار پیچیده تر که دارای چنین فیلتری هستند، استفاده شود. برخلاف ترانسفورماتورها یا سایر اجزاء سیستم های قدرت، که ممکن است شبیه به بلوک های ابتدایی شبکه قدرت باشند، یک *DSL* ابتدایی است فقط بوسیله بلوک دیاگرام های پیچیده تر ارجاع داده می‌شود و علاوه براین می تواند دریک زمان در بیش از یک مدل *DSL* پیچیده نیز استفاده گردد.

عناصر تعریف بلوک پیچیده را می‌توان با "*Grid Folder*" (پوشه های شبکه) در درخت پایگاه داده های *DIgSILENT* مقایسه کرد. آنها از طریق تعریف گرافیکی یک بلوک دیاگرام کنترل کننده که در آن اطلاعات گرافیکی و تمام قسمتهای منطقی را ذخیره می‌کند، مشخص می گردند. این قسمتها شامل سیگنال ها، مولفه های استاندارد کوچک (آدرس، ضرب کننده ها و غیره) یا *DSL* ابتدایی می‌باشند.

گرچه یک عنصر تعریف بلوک پیچیده به صورت گرافیکی ایجاد می‌شود، این امکان نیز مهیاست تا سایر معادلات *DSL* را که وارد کردن آنها در روش گرافیکی دشوار است، جداگانه تعریف نمود.

در این جا به محیط های گرافیکی که در آنها یک بلوک دیاگرام پیچیده ساخته می‌شود، پرداخته نشده است. برای کسب اطلاعات بیشتر به *User's Manual* مراجعه کنید.

3.4 The DIgSILENT Simulation Language (DSL)

زبان 'DSL' برای برنامه ریزی مدلهایی برای کنترل کننده های الکتریکی و سایر مولفه های استفاده شده در شبکه های قدرت الکتریکی، استفاده شده است. مانند هر شبیه سازی و زبان برنامه نویسی دیگر، یک گرامرخاص برای فرمول نویسی مدل فراهم شده است.

این گرامر به ترتیب زیر توضیح داده شده است :

أ : اصطلاحات و اختصارات

ب : گرامر عمومی *DSL*

ج : متغیرهای *DSL*

د : ساختار *DSL*

ه : گرامر *DSL*

و : پیش فرض ها

۳-٤-۱ اصطلاحات و اختصارات

اصطلاحات و اختصارات زیر برای توصیف گرامر *DSL* استفاده شده اند :

expr : عبارت محاسباتی ریاضی که نباید به علامت ';' ختم شود.

- عملگرهای محاسباتی / , * , - , + .
- ثوابت : همه اعدادی که به عنوان اعداد حقیقی عمل می‌کنند.
- توابع استاندارد :

sin (x) , cos (x) , tan (x) , asin (x) , acos (x) , atan (x) , sinh(x) , cosh(x) , tanh (x) , exp(x) , In(x) , log(x) [basis 10] , sqrt(x) [square root] , sqr(x) [power of 2] , pow (x , y), abs (x) , min (x,y) , max(x,y) , modulo(x , y) , trunc(x) , frac(x) , round(x) , ceil(x), floor (x).

این توابع استاندارد بطور مفصل در بخش بعدی تشریح شده اند.

- پرانتزها : (عبارات محاسباتی)

همه توابع مثلثاتی بر اساس رادیان ها می باشند (*RAD*).

مثال : *A=x1+2.45*T1/sin(3.14*y)*

boolexpr : عبارت منطقی که نباید به علامت ';' ختم شود.

- روابط منطقی : = , =< , >= , < > (inequality) , > , < .
- عملکردهای یکانی : . not
- عملکردهای باینری (دوتایی) : .eor. .nor. .nand. .or. .and.
- پرانتزها : (عبارت منطقی).

مثال : $A = \{ x1 > 0.\ .and.\ .\ not.\ x2 \leq 0.7 \}.or.T1=0.0$

string : هر چیز که بین علامت ' ' قرار گیرد.

مثال : *A = ' this is a string'*

۳-٤-۲ گرامر عمومی DSL

Line Length (طول خط) : حداکثر طول خط ۸۰ کاراکتر می‌باشد. خطوط طولانی تر را باید با استفاده از علامت & در ستون اول به دو یا چند بخش شکست و تقسیم نمود. علامت & در ابتدای ستون باعث می گردد تا سطر جاری و سطر پیشین به یکدیگر متصل گردند.

مثال :

```
y = lapprox ( x , 1.674 , 7.367 , 2.485 , 12.479 , 5.457 , 18.578
&                 6.783 , 15.54 , 8.453 , 12.589 , 9 , 569 , 6.478 )
```

قاعده شکستن خط را نمی‌توان درخصوص نامها یا رشته ها استفاده کرد.

Case sensitivity (حساسیت به کوچکی و بزرگی حروف) : تمام واژه های کلیدی، نامها، توابع، متغیرها، مدلها، ماکروها و. .. نسبت به حروف حساس هستند.

Blanks (جای خالی) : وقتی کد *DSL* پردازش می‌شود، تمام جاهای خالی برداشته می‌شوند. بجز در رشته ها که این جاهای خالی حفظ می‌شوند.

Comments (توضیحات) : علامت ! باعث می‌شود تا محتویات سطری که این علامت را دارد به عنوان توضیح منظور شود. وقتی کد *DSL* پردازش شود، تمام توضیحات برداشته خواهند شد. مثال :

```
! comments may start at the beginning of a line
x1. = select (at < > 0 , ! comments may be used in broken lines
&      (1 - sqr (x1) / sqr (at) / Tw , 0 )
```

۳-٤-۳ متغیرهای DSL

یک مدل *DSL* می‌تواند از ۵ نوع متغیر استفاده کند :

Output signals (سیگنال های خروجی) : در مدلهای پیچیده تر از سیگنال های خروجی به عنوان ورودی استفاده می شود.

Input Signals (سیگنال های ورودی) : متغیرهای ورودی ممکن است از مدلهای دیگر *DSL* یا از یک عنصر سیستم قدرت بدست آمده باشند. در مورد دوم، می توان از ولتاژها و جریان ها، و هر سیگنال موجود دیگری که از آنالیز شبکه قدرت بوجود آمده باشد، در مدل *DSL* استفاده نمود.

State variables : (متغیرهای حالت) : متغیرهای حالت، سیگنال های وابسته به زمانی می باشند که در خود مدل *DSL* ایجاد شده و مورد استفاده قرار می گیرند.

Parameters (پارامترها) : پارامترها اعداد فقط خواندنی هستند که به منظور تغییردادن رفتار مدل *DSL* تنظیم می‌شوند.

Internal variables (متغیرهای داخلی) : متغیرهای داخلی تعریف شده و در مدل *DSL* استفاده شده اند و سبب تسهیل در ساختن مجموعه معادلات *DSL* می‌شوند.

نکات :

- یک متغیر حالت نمی تواند در یک زمان به عنوان متغیر خروجی هم باشد؛ اگر چنین امری لازم باشد استفاده از یک متغیر جانشین مثل y = x1 توصیه می‌شود.
- همه پارامترها اعداد واقعی هستند.
- یک پارامتر خاص ' آرایه – iiii ' (با حداکثر ٤ رقم i)، با تعداد 2*iiii عنصر به منظور تعریف منحنی ها در دسترس می باشد (به روش "lapprox" نگاه کنید).
- تنها به مشتق های متغیرهای حالت می‌توان عبارتی را اختصاص داد.

۳-٤-٤ ساختار DSL

مدلهای *DSL* از سه قسمت ساخته شده اند :

- قسمت واسطه، که نام مدل، عنوان، طبقه بندی و مجموعه متغیرها را بیان می‌کند. این قسمت در صفحه اول کادر محاوره ای ویرایش بلوک دیاگرام تنظیم شده است.
- کد تعریف
- کد معادله

کد معادله و کد تعریف، شبکه کنترل کننده واقعی را به وجود آورده و در بخش های بعدی مورد بررسی قرار خواهند گرفت.

۳-٤-۵ کد تعریف

یک ***کد تعریف*** در قسمت معادله یک مدل *DSL* برای تعریف مشخصه های پارامتر و شرایط اولیه استفاده می‌شود.

واحد و توصیف

vardef (varnm) = unitstring;namestring واحد و نام برای متغیر *varnm*

مثالها :

vardef (Ton) = 's' ; ' pick up time for restart ' ! defines unit and name
vardef (Ton) = ; ' pick up time for restart ' ! only defines name.
vardef (Ton) = ' s ' ; ! only defines unit

[varnm] = unitstring : واحد برای متغیر *varnm*، حداکثرتا پهنای ۱۰ کاراکتر.

مثال :

[Ton] = 's' واحد را تعریف می کند!

تذکر : اگر واحد متغیرهای جانشین (جایگذاری شده) با واحدهای تعریف شده مطابقت نداشته باشند، بوسیله یک ماکرو فراخوانده شده می توان پیغام های خطا را ایجاد نمود.

محدوده مقادیر معتبر

Limits (varnm) = [/(minimum value , maximum value] /) : فاصله مجاز برای متغیر varnm را تعریف می کند. نوسانات حدود فاصله در طی عمل شبیه سازی گزارش خواهد شد :

limits (yt) = (,1)

هم ارز است با

output (yt > 1 ,' Maximum exceeded: yt = yt>1')

پرانتزهای "(" و ")" در هر سمت محدوده تعریف فاصله که قرار گیرند آن حد (مقدار حداقل یا حداکثر) را مستثنی می‌کنند. ودر مقابل براکتهای ']' و '[' در برگیرنده مقادیر کرانه ای فاصله خواهند بود.

مثالها :

limits (x) = [min , max] ! $min \leq x \leq max$
limits (x) = [min , max] ! $min < x \leq max$
limits (x) = (, max] ! $x \leq max$
limits (x) = (min ,) ! $min < x$

اگـر لازم و ممکـن باشـد، برنامـه بطورخودکـار کـوچکترین فاصـله را در بـین چنـدین فاصـله از همـان متغیرها را تعیین می‌کند. برای مثال : limits(x)=(1,3) و limits(x)=(2,4] به معنی $2<x<3$.

مـدلهای مـاکرو اغلـب حـدود متغیرهـای معینـی را تعریـف مـی‌نماینـد. مـدلی کـه از مـاکرو اسـتفاده مـی‌کنـد همچنین ممکن است حـدود متغیرهـایی را کـه در فراخـوانی هـای مـاکرو اسـتفاده شـده انـد را نیـز تعریـف کنـد. روش "*Smallest interval*" (کـوچکترین فاصـله) بـه مـدل فراخوانـده شـده ایـن اجـازه را مـی داد کـه حـدود پارامتر را بدون ایجاد اختلال در تعاریف حدود ماکروی داخلی، مجدداً تعیین نماید.

شرایط اولیه

تنظیم مستقیم

Inc(varnm) = expr : تعریـف شـرایط اولیـه متغیـر *varnm*. اگـر inc(varnm) تعریـف نـشده باشـد، عبـارت انتـسابی عـادی محاسـبه خواهـد شـد. (ایـن کـار فقـط وقتـی ممکـن اسـت کـه *varnm* از نـوع ورودی یـا داخلـی باشـد). اگـر (incvarnm) تعریـف شـده باشـد، وقتـی مـدل دوبـاره راه انـدازی مـی‌شـود، مجدداً ارزیابی خواهد شد.

Inco(varnm) = expr تعریـف شـرایط اولیـه متغیـر *varnm*، بـرای متغیرهـای ورودی و خروجـی وصل نشده مـی باشـد. ایـن نـوع از دسـتور *inc()* فقـط وقتـی اسـتفاده مـی شـود کـه متغیـر *varnm* نتوانـد از طریـق شـرایط اولیـه سـیگنال خروجـی و ورودی متـصل، راه انـدازی شـود. بنـابراین دسـتور*inc()* بـرای بـاز کردن احتمالی ترمینالهای ورودی و خروجی استفاده می‌شود.

incfix(varnm) = expr ایـن گونـه از دسـتور *inc ()* فقـط وقتـی اعتبـار دارد کـه مقـداردهی اولیـه خودکار باشد معتبر می‌باشد و بـرای تعیـین مقـادیر اولیـه در موقعیـت هـای دوگانـه اسـتفاده مـی‌شـود. بـا *incfix*، یک یا چند متغیر مستقیماً مقداردهی اولیه شـده درحـالی کـه مقـادیر اولیـه سـایر متغیرهـا مـی‌تواننـد بطورخودکـار مقداردهی شوند.

مثال :

مـدل *AVR* دارای دو ورودی(upss, usetp) و یـک خروجـی (uerrs) مـی‌باشـد. بـا مقـدار خروجـی تنها، نمی‌توان بطور همزمـان و خودکـار بـه هـر دو ورودی مقـدار اولیـه نـسبت داد. بنـابراین مقـدار اولیـه یکـی از ورودیهـا را بایـد عـدد ثـابتی داد، بـه طـور مثـال 'incfix (upss)=0' آنگـاه مقـدار اولیـه *usetp* بـا اسـتفاده از *upss = 0* بطورخودکار تعیین می‌شود.

تنظیم تکرار

بـه منظـور تعیـین مقـادیر اولیـه بطـورمکرر، سـه تـابع وجـود دارد کـه عبارتنـد از : *newtonnic, intervalinc, loopinc*. ایـن توابـع بـرای پیـدا کـردن مقـدار اولیـه یـک مجموعـه از پارامترهـا استفاده می‌شود، اگـر مقـادیر اولیـه مجموعـه دیگـر پارامترهـا کـه درواقـع تـوابعی از مجموعـه اول هـستند، معلـوم

باشند. توابع تکرار برای پیدا کردن (تقریبی) مقادیر پارامترهای نا مشخصی که این پارامتر شناخته شده مقادیر اولیه آنها را می‌گیرد، استفاده می‌شوند.

loopinc(varnm, min, max, step , eps) : این تابع یک جستجوی خطی ساده را برای یافتن مقدار منحصر بفردی انجام می‌دهد که در ازاء آن پارامتر *varnm* نزدیکترین مقدار به مقدار اولیه معلوم خود داشته باشد.

Varnm = متغیر نهایی، که مقدار اولیه معلوم باشد.

حد بالا = max حد پایین = min

حداکثر خطا = eps اندازه گام = step

مثال : inc(a) = loopinc(b , -5 , 5 , 0.01 , 0.001) یعنی :

مقدار اولیه متغیر "a" با محاسبه پارامتر "b" بدست می‌آید، از a=-5 شروع شده و در a=5 پایان می‌یابد و اندازه گام افزایشی ۰/۰۱ می‌باشد.

مقدار برگشتی : مقداری برای متغیر "a" خواهد بود که در ازاء کوچکترین انحراف متغیر "b" از مقدار اولیه اش، بدست آمده است. اگر کمترین انحراف از ماکزیمم خطای *eps* بزرگتر باشد، هشداری داده خواهد شد.

محدودیت : فقط در سمت راست دستور inc( ) می‌تواند استفاده شود.

intervalinc(varnm, min, max, iter, eps) : این تابع یک جستجوی بین فاصله ای (با تقسیم فاصله به اجزاء ریزتر) را برای یافتن مقدار منحصر بفردی انجام می‌دهد که در ازاء آن پارامتر *varnm* نزدیکترین مقدار به مقدار اولیه معلوم خود داشته باشد.

Varnm = متغیر نهایی، که مقدار اولیه معلوم باشد.

حداکثر تعداد دفعات تکرار= iter، حد بالا = max، حد پایین= min، حداکثر خطا = expr4

مثال : inc(a) = intervalinc(b , -5 , 5 , 40 ,0.001)

توضیح : از طریق تقسیم متوالی فاصله [۵ و ۵-] به فواصل کوچکتر سعی دارد تا مقدار اولیه متغیر "a" را بیابد و تا زمانی که انحراف متغیر "b" با مقدار اولیه اش از *eps* کمتر باشد، عملیات ادامه خواهد یافت. اگر تعداد تکرارها به حداکثر مقدار خود برسد، فرآیند تکرار متوقف خواهد شد و اگر کمترین انحراف بیش از *eps* باشد یک پیغام هشدار داده می‌شود.

محدودیت : فقط در سمت راست دستور inc( ) می‌تواند استفاده شود.

newtoninc(initexpr , start , iter , eps) : با هدف دستیابی به یک یا چند پارامتر، یک جستجوی تکراری نیوتن را از طریق حداقل کردن خطا در یک مجموعه از زوج معادلات انجام می‌دهد.

intexpr = عبارتی که باید با پارامترهایی که مقدار اولیه آنها جستجو شده است، برابر باشد.

Start = مقدار راه انداز (اولیه) برای پارامتری که مقدار اولیه آن جستجو شده است.

iter = حداکثر تعداد مجاز تکرارها.

eps = حداکثر خطای مطلق مجاز بین *initexpr* و پارامتری که مقدار اولیه آن جستجو شده است.

مثال :

qto = 0.5
eps = 0.000001
maxiter = 100

inc (hedr) = newtoninc (hw - sqr (qedr) * (Rds + Rdr),hw,maxiter,eps)
inc (qt1) = newtoninc (pt1 / (4*dh * eta1) , qto , maxiter , eps)
inc (qt2) = newtoninc (pt2 / (4*dh * eta2) , qto , maxiter , eps)
inc (qt3) = newtoninc (pt3 / (4*dh * eta3) , qto , maxiter , eps)
inc (qt4) = newtoninc (pt4 / (4*dh * eta4) , qto , maxiter , eps)

این مثال قسمتی از تعاریف مقدار اولیه را برای مدلی نشان می‌دهد که در آن مقادیر اولیه ۵ پارامتر ('hedr','qt1',...,'qt4') با تشکیل سیستمی از زوج معادلات وحل آن با روش تکراری نیوتن، بطورهمزمان حل شده است. در نهایت خواهیم داشت:

hedr ≈ hω - sqr (qedr) * (Rds + Rdr)
qt1 ≈ pt1 / (4*dh * eta1)
qt2 ≈ pt2 / (4* dh * eta2)
qt3 ≈ pt3 / (4 * dh * eta 3)
qt4 ≈ pt4 / (4 * dh * eta4)

۳-٤-٦ کد معادله

در کد معادله، همه معادلات لازم برای ساختن مدلهای شبیه سازی گنجانده شده است. مجموعه معادلات، دسته ای از زوج معادلات دیفرانسیل را تعریف می‌کند که توصیف کننده توابع تبدیل بین سیگنالهای ورودی و خروجی است. توابع تبدیل می‌توانند از توابع ساده خطی با یک سیگنال خروجی و یک سیگنال ورودی تا توابع غیرخطی پیچیده ناپیوسته با چند ورودی و خروجی، متغیر باشند.

DSL برای توصیف ارتباط مستقیم بین سیگنالها و دیگر متغیرها استفاده می‌شود. ممکن است عبارات به خود متغیر یا به مشتق اول متغیر حالت آن اختصاص داده شوند. بنابراین معادلات دیفرانسیل با درجه بالاتر را باید با معرفی متغیرهای حالت دیگر به دسته ای از معادلات دیفرانسیل مرتبه اول تبدیل نمود.

۳-٤-۷ دستورمعادله

دستورات معادله برای اختصاص دادن فرمولها به پارامترها استفاده می‌شود، بنابراین ارتباط تمام پارامترها را در یک دسته از معادلات دیفرانسیل نشان می دهد.

گرامر :

Varnm = expr : فرمول expr را به متغیر varnm اختصاصی می‌دهد.

مثالها :

y = sin (a) + 3 x1*
y =. not. x1 >2.or.a<= 3

varnm. = expr : فرمول expr را به اولین مشتق متغیر varnm اختصاص می‌دهد.

مثالها :

x1. = (xe - x1) /T1
x2. = x1

توضیحات :

- معادل قرار دادن *DSL* در هر توالی ممکن است رخ دهد. این ترتیب و توالی بر محاسبه فرمول ها تأثیری ندارد.
- همه متغیرها از نوع اعشاری هستند، حتی اگر به یک عبارت دودویی تخصیص داده شده باشند، که در آن حالت مقدارش 0.000 یا 1.000 خواهد بود.
- وقتی یک متغیر z در یک عبارت منطقی استفاده می‌شود (مثلاً y =. not. z)، آنگاه تعبیر عدد منطقی 1 از z با توجه به رابطه (z > 0.5) بوده و به شکل زیر تعبیر می گردد :

y1 =. not. z معادل است با y1 = (z = < 0.5) .

- به هنگام ترکیب متغیرهای منطقی و ناگسسته در فرمول ها، اخطاری داده نمی شود بنابراین ادغام روابط زیر :

```
y =.  not. x1>2.or.a<= 3
z = 4.0* y + x1
```

منجر به هیچ پیغامی نخواهد شد : بسته به y ، z مقدار x1+4/0 یا فقط x1 را خواهد گرفت.

- اختصاص دادن مقدار به یک متغیر به گونه ای است که ارتباطات بین متغیرها تشخیص داده خواهد شد. برای مثال زیر :

```
1.  a = b+5
2.  b = x1
3.  x1.  = 1
```

ابتدا سطر دوم محاسبه شده و سپس از نتیجه بدست آمده آن، سطر اول محاسبه خواهد گردید.

- تشکیل حلقه از عبارات ریاضی مجاز نمی باشد. در مثال زیر یک پیغام خطا ظاهر خواهد شد :

```
a = b + 5
b = 2*a
```

- اگر هیچ مقداری به متغیر *varnm* نسبت داده نشود، *varnm* مقدار اولیه خود را حفظ خواهد کرد.
- عبارت سمت راست نباید شامل مشتق های عبارات باشد، مشتق ها باید فقط در سمت چپ معادله ظاهر شوند :

a = sin (x1.) : غلط و x1. = asin (a) : صحیح است.

ماکروهای DSL

یک ماکروی *DSL* مدل از پیش تعریف شده، تابع ابتدایی یا تابع پیچیده *DSL* است، که منظور از ابتدایی این است که باید در مدلهای سطح بالاتر *DSL* گنجانده شود. کادر محاوره ای ویرایش بلوک یک گزینه طبقه بندی "Macro" را ارائه می‌کند که می‌تواند در تنظیم علامت گذاری مدل به عنوان ماکرو به کار رود. لحاظ نمودن یک ماکروی *DSL* در یک مدل سطح بالاتر، یا با ایجاد یک مرجع بلوکی درگرافیکهای بلوک دیاگرام امکان پذیر است یا با ارتباط واضح و منطقی در یک معادله *DSL*.

گرامر :

varnm1,varnm2,...= macroname(i1,i2,... ;s1,s2,... ;p1,p2,... ;i1,i2,...)

ماکروی *DSL* با نام macroname سیگنال های خروجی را به متغیرهای varnm1، varnm2 و. .. و سیگنال های ورودی را به متغیرهای i1 و i2 اختصاص داده است. این ماکرو از متغیرهای حالت 's2' و 'p1'، پارامترهای 'p1' و 'p2'، و متغیرهای ورودی i1 و i2 نیز استفاده نموده است.

مثال : P1,P2 = '\ User\ I.BlkDef'(i1,i2;s1,s2;T1,T2)

در این مثال به P1 و P2 خروجی مدل 'User\ I.BlkDef' از *DSL* تخصیص داده شده است.

امکان فراخوانی های ماکرو در داخل فرمولها وجود ندارد، حتی اگر آنها فقط دارای یک متغیر خروجی باشند :

y = my_macro (x1 , s1 , p1 , i1) : مثالی صحیح می باشد،

اما

y = 3* my_macro (x1 , s1 , p1 , i1) + 4 نادرست است که باید به صورت زیر جایگزین شود :

y1 = my - macro (x1 , s1 , p1 , i1)

y = 3 * y1 + 4

کارکردن با ماکروی داخلی DSL

یک پیش تحلیلگر وظیفه دارد تا هر فراخوانی ماکرو را، با کد معادله ماکرو جانشین کند. متغیرهای مدل *DSL* ماکرو بوسیله متغیرهای استفاده شده در فراخوانی ماکرو، جایگزین شده اند. نامهای بومی متغیر ماکروها بعد از مرحله پیش تحلیلی، از بین خواهند رفت.

مدلهای DSL

به طورکلی، دو نوع مدل اصلی *DSL* وجود دارد که عبارتند از :

۱- مدلهای دستگاههای الکتریکی مانند ژنراتورها، بارها یا سیستم های *HVDC*. این مدلها بوسیله سیگنال خروجی اصلی خودشان 'complex device current' که در یک باس بار معین به شبکه الکتریکی ترزیق شده است، مشخص شده اند. به هرحال، علاوه بر جریان های ادوات الکتریکی ممکن است هر متغیر دیگر را نیز به عنوان سیگنال خروجی تعریف نمود.

توجه : این ادوات سیستم قدرت که توسط کاربر تعریف شده اند، زمانی که این کتاب راهنما نوشته می شده، در نرم افزار وجود نداشته است.

۲- مدلهایی با سیگنالهای خروجی که مستقیماً به شبکه الکتریکی(دستگاههای کلی) تزریق نشده اند. از جمله این نوع مدلها واحدهای محرک اولیه، کنترل کننده های ولتاژ، رله ها، روشهای محاسبه و غیره را می توان نام برد.

حوادث و پیغامها

زبان *DSL* روش هایی را برای ایجاد یک رخداد وقفه یا برای ارسال یک پیغام خطا به پنجره خروجی فراهم نموده است که عبارتند از :

- روش Fault(boolexpr , event_string) که یک رخداد را ایجاد نموده و در آغاز هر پله زمانی، آن را محاسبه می‌کند.

- روش output(boolexpr , message_string) یک پیغام خروجی ایجاد می‌کند و در پایان هر پله زمانی محاسبه می‌شود.

روشهای ***fault*** و ***output*** در هر پله زمانی از فرایند شبیه‌سازی مدل، محاسبه می‌شوند. مرتبه اولی که مشخص شود ***boolexpr*** صحیح است، این رشته پردازش خواهد شد و پیغامی به پنجره خروجی فرستاده می‌شود یا رخدادی به صف حادثه ***DIgSILENT*** کشیده خواهد شد. بعد از این، روشهای ***fault*** و ***output*** به منظور جلوگیری از ارسال سیل آسای پیغامها وحوادث تا زمانی که مدل 'DSL' دوباره راه اندازی شود، غیرفعال خواهند شد.

هر دو روش به تفصیل در بخشهای بعدی تشریح شده اند.

Output(boolexpr , message_string)

message_string ممکن است شامل متغیرها و تابع خاص num(boolexpr) یا num(expr) باشد :

- اسامی متغیرهایی که بلافاصله بعد از علامت '=' ظاهر می‌شوند بوسیله مقادیر واقعی خودشان جانشین خواهند شد. برای مثال :

output(yymax : 'maximum excoeded : y = yt > ymax = ymax')

که ممکن است منجر به تولید پیغام " از مقدار حداکثر تجاوز کرده است : yt=1.2 > ymny=10" گردد.

- num(expr) یا num(bool expr) با مقدار محاسبه شده فرمول، جایگزین خواهند شد.

مثال :

value = num (a + b) که ممکن است مقدار value = 3.5000 را ایجاد نماید.

Fault(boolexpr , event_string)

هر مدل ***DSL*** دارای قابلیت انتقال حوادث به محلی جهت ذخیره سازی می‌باشد. به طور مثال در مدل ***DSL*** یک رله دیستانس، می‌تواند کلید قدرت یک خط را از طریق تعریف صحیح رخداد کلید باز نمود. "Throwing an event" (ایجاد یک رخداد) با اجرای یک رخداد موجود در پایگاه داده‌های ***DIgSILENT*** اجرا می‌شود.

در نتیجه، همه رخدادهایی که ممکن است بوسیله مدل ***DSL*** استفاده شده باشند باید بطورهمزمان با مدل ***DSL*** ایجاد گردند. همه آنها باید در داخل مدل مشترک ***ElmDSL***، دکمه ***event*** ذخیره شوند. بنابراین ***DSL events*** یک قسمت پیوسته از مدل ***DSL*** را تشکیل خواهند داد.

event_string درفرمول خطا بایستی به نام یکی از این رخدادها ارجاع داده شود. در محاسبه، اگر ***boolexpr*** درست باشد، آنگاه رخداد به درون پشته حادثه کشیده می‌شود. به محض این که فرایند شبیه سازی به این رخداد رسید، آن را اجرا خواهد نمود. در نتیجه یک رخداد تأخیری را ممکن است با تنظیم زمان اجرا جلوتر از زمان جاری و بوسیله مدل ***DSL*** اجرا نمود.

پارامترهای رخداد در این رشته خطا را می‌توان با اختصاص دادن یک مقدار جدید اصلاح کرد. روش کار همان است که در فوق و در مورد روش خروجی توصیف شد.

مثال :

Fault (u > 1.1 , 'name = MyswitchEvent1 dtime = 0.15 ')

اگـر متغیـر u از 1.1 بیـشتر گـردد، آنگـاه رخـداد نامیـده شـده بـا MySwitchEvent1 بـه درون پـشته رخـداد کـشیده خواهـد شـد و متغیـر *dtime* (زمـان نـسبی رخـداد) در ۱۵ میلـی ثانیـه تنظـیم خواهـد گردیـد. بنابراین، این رخداد به آن مقدار زمانی به تـأخیر خواهـد افتـاد، کـه در ایـن مـورد زمـان لازم بـرای بـاز کـردن یـک سـوییچ را شـبیه سـازی مـی‌کنـد. و سـوییچی کـه بـاز مـی‌شـود در رخـداد "MySwitchEvent1" تعریـف شده است.

۳-٤-۸ مثالها

توربین حرارتی دوباره گرم شده با بخار ذخیره شده

مدل کنترل کننده :

```
model pt , ptmw
'pmu - 1 ' ( at , sgn , cosn , ngnum ; x1 , x2 , x3 , x4 ;Thp ,  Tip , Tlp , alflp , Tspi )
          [T1] = 's'
   limits (T1) = [0 , )
limits (alfhp ) = [0 , 1 ]
vardef (alfhp) = ; 'High pressure turbine ratio ' ;
limits (alflp) = [0 , 1 - alfhp ]
vardef (alflp ) = ; ' Low pressure turbine ratio ' ;
vardef (Tspi) = 's' ; 'Boiler capacity time constant ' ;
limits (Tspi) = ( 0 , )
vardef (Thp ) = 's' ; 'High pressure turbine time constant ' ;
vardef (Tip) = 's' ; ' First reheater time constant ' ;
vardef (Tlp) = ' s ' ; ' second reheater time constant ' ;

        inc (x1) = y / k
       inc (xe ) = y / k
       inc (x4) = 1.0
       inc (at) = pt
inc (steamflow0) = pt
       inc (ylp ) = pt
   x1.     = select ( T1 > 0 , ( xe - x1) / T1.  0 )
   y      = k* select (T1 > 0 , x1 , xe )       ! if T1 = 0 = >  y = xe
steamflow = at * x4
    x4.  = ( steamflow0 - steamflow ) / Tspi     ! boiler
    yhp = PT1 (steamflow ; x1 ; Thp )     ! high  pressure part
     yip  = PT1 ( yhp ; x2 ; Tip )    ! medium pressure part
     ylp  = PT1 ( yip ; x3 ; Tlp )     ! low   pressure part
     pt  = yhp * alfhp + ylp * alflp + yip * ( 1.  0 - alfhp - alflp )
   ptmw  = pt * sgn * cosn * ngnum    !  only for output purposes
```

ماکروی استفاده شده 'PT1' به صورت زیر تعریف می‌شود :

```
model y = ' PT1 ' ( xe ; x1 ; k , T1 ; )
          x1.   = Select ( T1 > 0 , ( xe - x1) / T1 , 0)
          y   = k* select (T1 > 0 ,x1 , xe )          ! if T1 = 0   = >   y = xe
      inc (x1) = y/k
      inc (xe) = y / k
        [T1] = 's'
limits (T1) = [ 0 , )
```

3.5 DSL Funtions

۳-۵-۱ توابع استاندارد

sin(x)	sine	**sin(1.2)=0.93203**
cos(x)	cosine	**cos(1.2)=0.36236**
tan(x)	tangent	**tan(1.2)=2.57215**
asin(x)	arcsine	**asin(0.93203)=1.2**
acos(x)	arccosine	**acos(0.36236)=1.2**
atan(x)	arctangent	**atan(2.57215)=1.2**
sinh(x)	hyperbolic sine	**sinh(1.5708)=2.3013**
cosh(x)	hyperbolic cosine	**cosh(1.5708)=2.5092**
tanh(x)	hyperbolic tangent	**tanh(0.7616)=1.0000**
exp(x)	exponential value	**exp(1.0)=2.718281**
ln(x)	natural logarithm	**ln(2.718281)=1.0**
log(x)	log10	**log(100)=2**
sqrt(x)	square root	**sqrt(9.5)=3.0822**
sqr(x)	power of 2	**sqr(3.0822)=9.5**
pow (x,y)	power of y	**pow(2.5, 3.4)=22.5422**
abs(x)	absolute value	**abs(-2.34)=2.34**
min(x,y)	smaller value	**min(6.4, 1.5)=1.5**
max(x,y)	larger value	**max(6.4, 1.5)=6.4**
modulo(x,y)	remainder of x/y	**modulo(15.6, 3.4)=0.58823**
trunc(x)	integral part	**trunc(-4.58823)=-4.0000**
frac(x)	fractional part	**frac(-4.58823)=-0.58823**
round(x)	closest integer	**round(1.65)=2.000**
ceil(x)	smallest larger integer	**ceil(1.15)=2.000**
floor(x)	largest smaller integer	**floor(1.78) = 1.000**
pi()	3.141592**...**	**p=3.141592...**
twopi()	6.283185**...**	**2p=6.283185...**
e()	2,718281**...**	**e=2,718281...**

آرگومان همه توابع مثلثاتی رادیان می‌باشد.

۳-۵-۲ توابع خاص

Lim(u , min , max) = تابع محدود کننده غیر خطی.

$$returns \begin{cases} \min & ifx < \min \\ \max & ifx > \max \\ xif & \min \le x \le \max \end{cases}$$

Limstate (x1 , min , max) = تابع محدود کننده غیر خطی برای ایجاد انتگرال گیرهای محدود شده.

مثال :

```
x1. = xe/Ti ;
y  = limstate (x1 , min , max ) ;
```

این مورد قبلاً بوسیله استفاده از توابع " Selecte" و " lim " بررسی شده است.

```
x1. & = & select ( { x1$>$ = max.  and.  xe$ > $0}
&          . or.  { x1 $ < $ = min.  and.  xe$ < $0 } , 0 , xe / Ti ) ;
y    & = &  lim (x1 , min , max ) ;
```

delay(x , Tdelay) : تابع تأخیری. مقدار x(Tnow) را ذخیره کرده و مقدار x(Tnow,Tdelay) را برمی‌گرداند. *Tdelay* برحسب ثانیه بوده و از 0.0 بزرگتر می‌باشد. *Tdelay* باید به صورت یک ضریب مستقل از زمان محاسبه شود و بنابراین باید فقط شامل ضرایب ثابت و متغیرهای پارامتری باشد. عبارت *x(t)* ممکن است شامل توابع دیگری باشد.

مثال : y = delay(xe + delay (x1 , 1.0) , 2.0)

تنظیم مجدد یک مدل *DSL* توابع تأخیری آن را با *x(Treset)* مقداردهی اولیه می‌نماید.

selecte(boolepr , x , y) : اگر *boolexpr* درست باشد *x* را برمی‌گرداند و درغیر این صورت *y* برگردانده می‌شود.

مثال : برای جلوگیری از تقسیم شدن بر صفر

```
x1.  = select ( T1 > 0 , xe / T1 , o/o )
```

Lapprox(x , array_iiii) : تقریب خطی y = f (x) را در جایی که f بوسیله ***array_iiii*** تعریف شده است، برمی‌گرداند.

مثال :

```
y = lapprox ( 1.8 , array  - valve )
```

invlapprox(y , array - iiii) : تابع Lapprox را با ***array_iiii*** معکوس می‌کند.

file(ascil-parm , expr)!OBSOLETE : از یک عنصر *ElmFile* به جای مدل مرکب استفاده می‌کند.

Picdro(boolexpr , Tpick , Tdrop) : تابع منطقی 'pick - up - drop - off' برای رله ها مفید می‌باشد. این تابع حالت منطقی داخلی را بر می‌گرداند یعنی : صفر یا یک. حالت داخلی عبارت است از :

- اگـر بـرای مـدت زمـانی معـادل حـداقل *Tpick* ثانیـه، 1 = boolexpr باشـد، از صـفر بـه یـک تغییر پیدا می‌کند.
- اگر پس از *Tdrop* ثانیه، 0 = bollexpr باشد، از یک به صفر تغییر پیدا می‌کند.
- در وضعیت های دیگر بدون تغییر باقی می‌ماند.

flip flop(boolset , boolreset) : تـابع فلیـپ فـلاپ منطقـی. ایـن تـابع حالـت منطقـی داخلـی را بر می‌گرداند یعنی : صفر یا یک. حالت داخلی عبارت است از:

- اگر *boolset = 1* و *boolreset = 0* باشد. از صفر به یک تغییر می‌یابد(SET).
- اگر *boolreset = 1* و *boolset = 0* باشد. از یک به صفرتغییرمی‌یابد(RESET).
- در شرایط دیگر بدون تغییر باقی می‌ماند(HOLD).

Boolset : مقدار اولیه

شرایط اولیه *boolset = boolreset = 1* باعث ایجاد پیغام خطا می‌شود.

aflipflop(x , boolset , boolreset) : یـک تـابع فلیـپ فـلاپ پیوسـته. اگـر وضـعیت داخلـی مـساوی ۱ باشـد مقـدار (قبلـی) را بـرای x در SET-time برگردانـده و در غیـر ایـن صـورت مقـدار کنـونی x را بر می‌گرداند.

این وضعیت داخلی عبارت است از :

- اگر ‘ *boolset = 1* و *boolreset = 0*’ باشد. از صفر به یک تغییر می‌یابد(SET).
- اگر ‘ *boolset = 0* و*boolrest = 1*’ باشد. از یک به صفر تغییر می‌یابد(RESET).
- در موقعیت های دیگر بدون تغییر باقی می‌ماند(HOLD).

Time() : زمان شبیه سازی جاری را بر می‌گرداند.

مثال :

t = time () , y = sin (t) or y = sin (time ( ))