```
% Os dois primeiros exemplos são testes de conceito.
```

Conexões não implementadas

Ainda não temos nenhuma conexão diferente de TCP.

Reserva para em caso no futuro seja desejável outras formas de conexão.

E para ajudar a desacoplar e permitir sobrecarga de métodos.

```
% try
%     Analysers.Analyser.connGPIB('1234')
% catch exception
%     disp(exception.message)
% end
```

Execução com o simulador SA2500PC

Alocação dinâmica ainda não implementada (perfumaria prevista):

Caso só passe o IP, teríamos uma opção de auto discovery:

```
% % Auto discovery
% try
%     Analysers.Analyser.connTCP("localhost");
% catch exception
%     disp(exception.message)
% end
%
% % Ref. portas a varrer:
% % 5025  - Keysight e R&S
% % 5555  - R&S EB500
% % 9001  - Anritsu
% % 34835 - Tektronix
```

Instrumento simulado (Download em SA2500PC).

Conectado e respondendo à sua identificação:

```
% disp('Propriedades:')
% callTCP = Analysers.Analyser.connTCP("localhost", 34835) % Simulador
callTCP = Analysers.Analyser.connTCP("192.168.48.2", 34835)
```

```
callTCP =
  dictionary (string  string) with 6 entries:

    "Factory"  "TEKTRONIX"
    "model"    "SA2500"
    "serial"   "B040211"
    "version"  "7.050"
```

```
"ip"      "192.168.48.2"
"port"    "34835"
```

Instância dinâmica.

Cada fabricante deve ter, na pasta 'Analysers', sua classe como mesmo nome de sua superclasse (prop:Factory), e cada especifidade de um certo modelo (prop:model) deve estar com o mesmo nome, o que permite escalonar e isolar os componentes em uma interface unificada, e granularizada para o serviço esperado.

O 'Analyser' deve conter todos os comandos genéricos da SCPI (Standard Commands for Programmable Instruments) e IEEE 488.2 Common Commands, este último inicia por um asterisco..

Com base na IDN que o instrumento responde, a instância sempre herda todos os comandos do 'Analyser'.

No caso, o "is a" está representado no classdef como "classdef TEKTRONIX < Analyser", ou seja, Um Tektronix é um analisador, e o SP2500 é um Tektronx ("classdef SA2500PC < TEKTRONIX")

Neste caso herda os comuns e sobrecarrega os comandos do Tektronix, e os específicos do modelo SA2500PC:

```
disp('Instancia Classes:')
```

```
Instancia Classes:
```

```
obj = Analysers.Analyser.instance(callTCP);
```

```
    "Analyer: Base de comando do fabricante"    "TEKTRONIX"
```

O simulador fecha quando recebe o comando de reset, então só fingimos o reset para evitar isso (na implementação específica do modelo). O que ainda oportuna outros casos de uso em casos diferentes, como o EB500 que precisa abrir uma porta específica para receber stream UDP.

Um instrumento real não terá o sufixo PC e portanto automaticamente não herdará esse método, o que possibilita compartilar comandos e especificar no modelo o que for diferente nele.

Aplicando uma sobrecarga no modelo SA2500PC com a resposta da classe:

```
obj.scpiReset;
```

```
Analyer.scpiReset: Criando nova conexão TCP.
```

Comandos gerais de inicialização:

```
obj.startUp()
```

```
TEKTRONIX: Start Ok.
```

```

```

Teste geral de conectividade (SCPI):

```
obj.ping()
```

```
Analyser.ping: Mesma conexão
Analyser.ping: Resposta IDN recebida:
TEKTRONIX,SA2500,B040211,7.050
```

```

```

Obtém parâmetros do objeto:

```
disp('getSpan:')
```

```
getSpan:
```

```
disp(obj.getSpan())
```

```
200000000
```

```
disp('Parâmetros:')
```

```
Parâmetros:
```

```
obj.getParms()
```

```
ans =
  dictionary (string  string) with 10 entries:

    "Function"  "NORM"
    "AVGCount"  "20"
    "Detection" "POS"
    "UnitPower" "DBM"
    "FStart"    "759000000"
    "FStop"     "959000000"
    "ResAuto"   "1"
    "Res"       "200000"
    "InputGain" "0"
    "Att"       "50"
```

```

```

Operação

Aqui perguntamos se o parâmetro corresponde ao que foi solicitado (assert). E verificamos as respostas no simulador em tempo real:

```
disp('Pausas para observar o comportamento:')
```

```
Pausas para observar o comportamento:
```

```
%pause(5) % tempo para ajuste manual de teste.
```

```
% % Testes
% obj.setFreq(120000000)
% obj.setSpan(50000)
% assert(str2double(obj.getSpan) == 50000, 'O Span não foi ajustado.')
% obj.setRes(2000)
% assert(str2double(obj.getRes) == 2000, 'O Span não foi ajustado.')
%
pause(1)
obj.setFreq(100000000)
obj.setSpan(10000)

% disp('Nova conexão simulada pelo ping (caso não responda):')
% obj.conn = [];

% Isso gera um alerta na janela do simulador,
% mas não retorna erro.
% try
%     RBW = obj.getRes();
%     assert(str2double(RBW) == 2000, 'A Resolução foi alterada automaticamente.')
% catch exception
%     disp('Resolução alterada automaticamente para:')
%     disp(RBW)
% end

pause(1)
disp('Ajuste em faixa larga')
```

```
Ajuste em faixa larga
```

```
obj.setFreq(88000000, 108000000)
```

```
obj.setRes(20000) % Warning: Data out of range
% try
%     assert(str2double(obj.getSpan) == 10000, 'O Span não foi ajustado.')
% catch
%     disp('Se atribuir o span depois gera erro "out of range"')
% end

% pause(1)
% % Observar que o com Spectrum o RBW Auto em verde
% obj.setRes('Auto')
% obj.setFreq(88000000, 108000000)

% Para observar a faixa em auto level
pause(3)
% Escolhe uma portadora FM para análise
freq = 100300000;

obj.setSpan(500000)
obj.setFreq(freq)
assert(str2double(obj.getSpan) == 500000, 'O Span não foi ajustado.')

% Cuidado! Os pisos não correspondem aos de um instrumento real!

%obj.preAmp('On')

% % Teste dos níveis de atenuação (comentado para poupar tempo)
% % Para observar abrir o menu Spectrum -> More -> Ampl
% for attstep = 5:5:50
%     obj.setAtt(attstep);
%     pause(0.5)
%
%     % O pré desativa sozinho com att acima de 15dB
%     % Mesmo expressamente solicitado, sem erro:
%     obj.preAmp('On')
% end
```

```
% Esse bloco try é para liberar a conexão em caso de erro.
try
    trace = obj.getTrace(1);

    disp('Trace:')



    % Só as 5 primeiras linhas
    disp(trace (1:5,:));
```

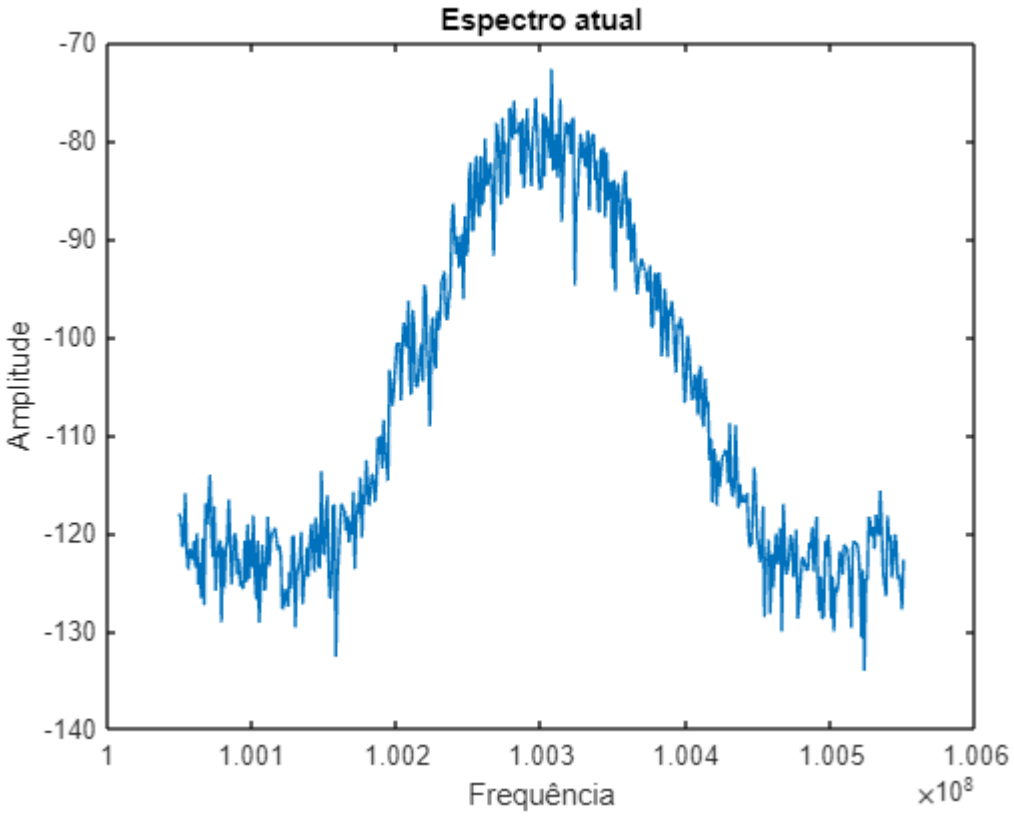
```
% Plota o Trace
p = plot(trace.freq,trace.value,"DisplayName","value");

% Add xlabel, ylabel, title, and legend
xlabel("Frequência")
ylabel("Amplitude")
title("Espectro atual")
legend('off');

disp('Leitura do marcador:')
nivel = obj.getMarker(freq, 1);
sprintf('Em %i MHz o nível é: %f', freq, nivel)
```

Trace:

freq	value
1.0005e+08	-118.07
1.0005e+08	-118.73
1.0005e+08	-121.49
1.0005e+08	-120.85
1.0005e+08	-115.97




```
Leitura do marcador:
ans =
'Em 100300000 MHz o nível é: -90.636063'
```

```
    disp('Adquirindo os traços com 3s de delay.');
```

```
Adquirindo os traços com 3s de delay.
```

```
    disp('Para amenizar problemas de sinc.')
```

```
Para amenizar problemas de sinc.
```

```
    trcs = fcn.getTracesFromUnit(obj, 10);
```

```
ans =
"Trace nº 1"
ans =
"Trace nº 2"
ans =
"Trace nº 3"
ans =
"Trace nº 4"
ans =
"Trace nº 5"
ans =
"Trace nº 6"
ans =
"Trace nº 7"
ans =
"Trace nº 8"
ans =
"Trace nº 9"
ans =
"Trace nº 10"
```

```
    disp("Total size:")
```

```
Total size:
```

```
    size(trcs)
```

```
ans = 1×2
    10   501
```

```
    disp(trcs)
```

```
 -120.5881 -127.6184 -118.2587 -117.9575 -122.1557 -118.3135 -123.8988 -131.7202 -127.8473 -126.8056 -122.3191 -121.
 -122.6561 -116.6455 -117.8263 -124.1106 -126.4947 -124.4544 -126.8409 -125.1574 -122.9766 -125.7751 -121.4911 -128.
 -118.1422 -120.3901 -123.8523 -124.6757 -125.8015 -125.1340 -113.1195 -122.0975 -120.2159 -120.4543 -118.8022 -122.
 -129.6135 -132.0400 -125.4900 -126.3960 -121.3681 -121.6488 -125.8365 -129.7543 -124.1538 -126.0825 -123.9454 -124.
 -127.9683 -123.1917 -123.0318 -120.8614 -118.3101 -125.1841 -127.6641 -130.7792 -124.9754 -124.8754 -119.1451 -119.
 -126.0678 -122.6031 -118.5860 -117.6811 -122.6659 -119.5536 -128.5413 -120.1085 -121.6778 -122.4699 -125.0641 -124.
 -116.7255 -126.6389 -126.8747 -123.0153 -126.4557 -126.9559 -118.6597 -121.2857 -118.8635 -123.0543 -126.3399 -125.
 -118.2525 -118.9576 -125.6776 -119.9893 -123.5357 -130.2115 -126.1444 -125.0722 -122.3521 -127.7672 -126.5277 -123.
 -126.7632 -127.6111 -127.4650 -124.2731 -125.3442 -123.8721 -121.6545 -126.7695 -123.9253 -128.7015 -120.1570 -126.
 -126.7242 -122.1654 -117.5293 -122.5510 -124.2927 -128.4210 -127.8852 -119.4087 -119.4293 -118.2958 -130.3887 -119.
```

```
catch exception
    % Encerra ativamente a conexão
    % Para liberar a porta em caso de erro
    obj.disconnect();
    error('TestTektronix: %s', exception.identifier);
```

```
end

% Se tudo der certo, encerra a conexão.
obj.disconnect()


disp("Pronto")
```

```
Pronto
```