```
% 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:
```

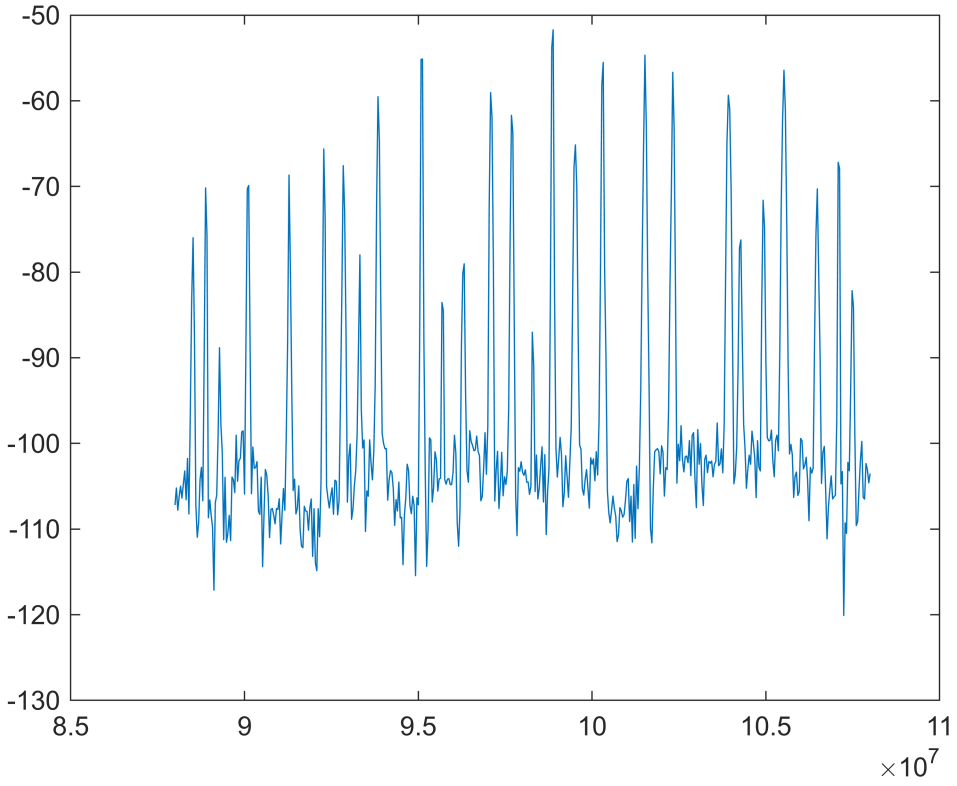
```
%pause(5) % tempo para ajuste manual de teste.
```

```
% % 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('Amostra em faixa larga')
```

```
Amostra em faixa larga
```

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

```
obj.setRes(20000) % Warning: Data out of range

trace = obj.getTrace(1);
plot(trace.freq,trace.value,"DisplayName","value");
```

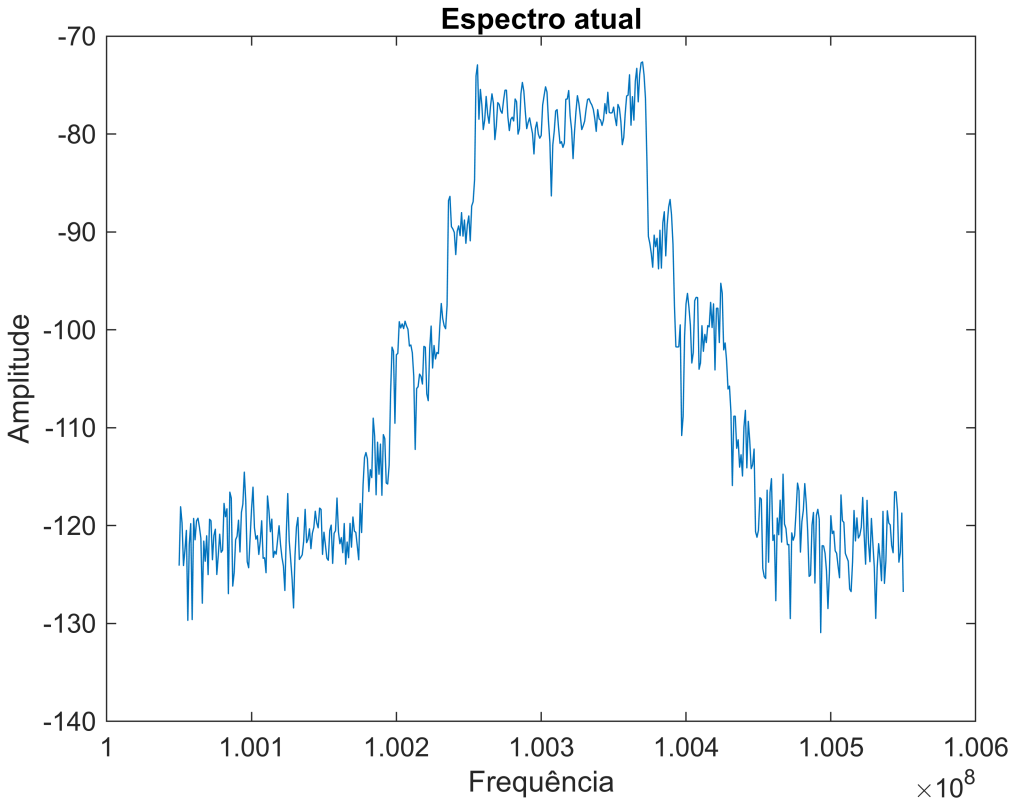



```
% try
%     assert(str2double(obj.getSpan) == 10000, 'O Span não foi ajustado.')
% catch
%     disp('Se atribuir o span depois gera erro "out of range"')
% end

% 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(2)
% 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');
```

```
Trace:
       freq         value
    _________    _______

    1.0005e+08    -124.09
    1.0005e+08    -118.07
    1.0005e+08    -119.72
    1.0005e+08    -124.08
    1.0005e+08    -122.49
```




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

```
ans =
"Trace 1"
ans =
"Trace 2"
ans =
"Trace 3"
Analyser: Aguardando sincronismo ...
ans =
"Trace 4"
ans =
"Trace 5"
Analyser: Aguardando sincronismo ...
ans =
"Trace 6"
Analyser: Aguardando sincronismo ...
ans =
"Trace 7"
ans =
"Trace 8"
ans =
"Trace 9"
ans =
"Trace 10"
```

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

```
Total size:
```

```
    size(trcs)
```

```
ans = 1×2
    10   501
```

```
    disp(trcs)
```

```
 -120.7582 -122.7377 -121.2379 -121.9706 -119.6026 -124.6685 -119.5793 -122.0503 -124.1646 -119.9926 -119.3859 -119
 -121.9494 -123.5384 -121.0253 -119.8317 -125.4829 -126.8641 -118.9766 -122.4644 -121.3633 -120.6636 -124.4111 -118
 -117.2874 -118.1633 -118.6291 -120.0673 -123.4269 -123.5825 -123.4198 -119.7804 -119.5396 -119.8766 -120.7034 -122
 -129.6998 -126.8783 -122.3551 -117.5253 -122.4178 -121.5663 -126.3359 -122.5400 -122.6017 -123.9044 -122.2319 -121
 -119.9453 -114.1979 -115.7262 -120.2507 -118.6526 -119.9375 -116.4623 -120.0326 -115.2786 -117.5014 -122.5925 -118
 -121.6370 -126.4135 -122.3119 -123.8988 -121.9370 -123.0631 -132.5995 -120.8661 -122.5046 -126.8545 -126.2529 -122
 -127.9329 -120.3792 -118.2625 -120.2984 -128.6878 -119.3894 -128.4121 -121.9607 -122.7194 -125.4676 -124.1557 -121
 -122.8870 -120.2384 -124.5661 -118.6248 -123.3525 -122.8461 -122.5882 -123.3896 -125.6984 -124.1229 -125.5988 -127
 -119.0552 -119.7251 -124.3407 -123.8298 -118.9086 -120.6468 -120.1693 -126.5284 -123.8059 -126.0098 -121.5700 -121
 -120.9598 -119.8867 -122.2069 -120.2082 -119.8009 -120.2621 -118.4552 -119.4334 -123.2394 -124.0115 -124.4401 -123
```

```
    % Um tempo para o analisador respirar
    % Porque eventualmente ele não responde
    pause(0.2)

    freq = 100300200;
    obj.setFreq(freq)
    disp('Leitura do marcador:')
```

```
Leitura do marcador:
```

```
    nivel = obj.getMarker(freq, 1);
    fprintf('Em %i MHz o nível é: %f\n', freq, nivel);
```

```
Em 100300200 MHz o nível é: -72.707726
```

```
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
```