```
% 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
```

```
Analyser: Conexão GPIB não implementada.
```

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
```

```
Analyser: Autodiscovery não implementado
```

```
%
% % 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('Observação do comportamento:')
```

```
Observação do comportamento:
```

```
% Esse bloco try é para liberar a conexão em caso de erro.
try
```

```
    % 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.disconnect();

    % 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
```

```
Analyer.getCMD: Criando nova conexão.
Resolução alterada automaticamente para:
1000
```

```
% pause(1)
disp('Amostra em faixa larga')
```

```
Amostra em faixa larga
```

```
obj.setFreq(88000000, 108000000)
obj.setRes(20000) % Warning: Data out of range

wtrace = obj.getTrace(1);

disp('Trace em faixa larga:')
```

```
Trace em faixa larga:
```

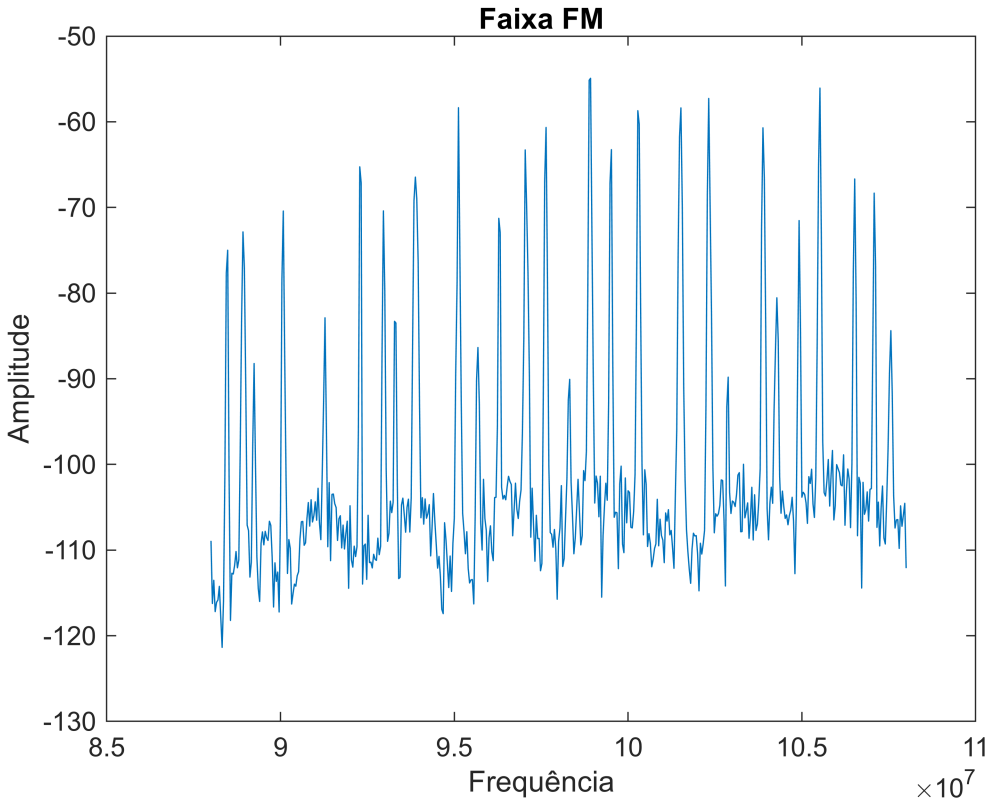
```
% Só as 10 primeiras linhas
disp(wtrace (1:10,:));
```

```
    freq        value
_________    _______

  8.8e+07    -108.92
8.804e+07    -116.24
8.808e+07    -113.52
8.812e+07    -117.19
8.816e+07     -116.1
 8.82e+07    -115.83
8.824e+07    -114.23
8.828e+07    -117.78
8.832e+07    -121.36
8.836e+07    -115.95
```

```
plot(wtrace.freq,wtrace.value,"DisplayName","value");
% drawnow;

% Add xlabel, ylabel, title, and legend
xlabel("Frequência");
ylabel("Amplitude");
title("Faixa FM");
```




```
legend('off');
% drawnow;

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

```
Se atribuir o span depois.
gera erro "out of range"
```

```
% 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.')

    obj.preAmp('On')

    % Teste dos níveis de atenuação
    % 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
```

```
    trace = obj.getTrace(1);

    disp('Trace para análise:')
```

```
Trace para análise:
```

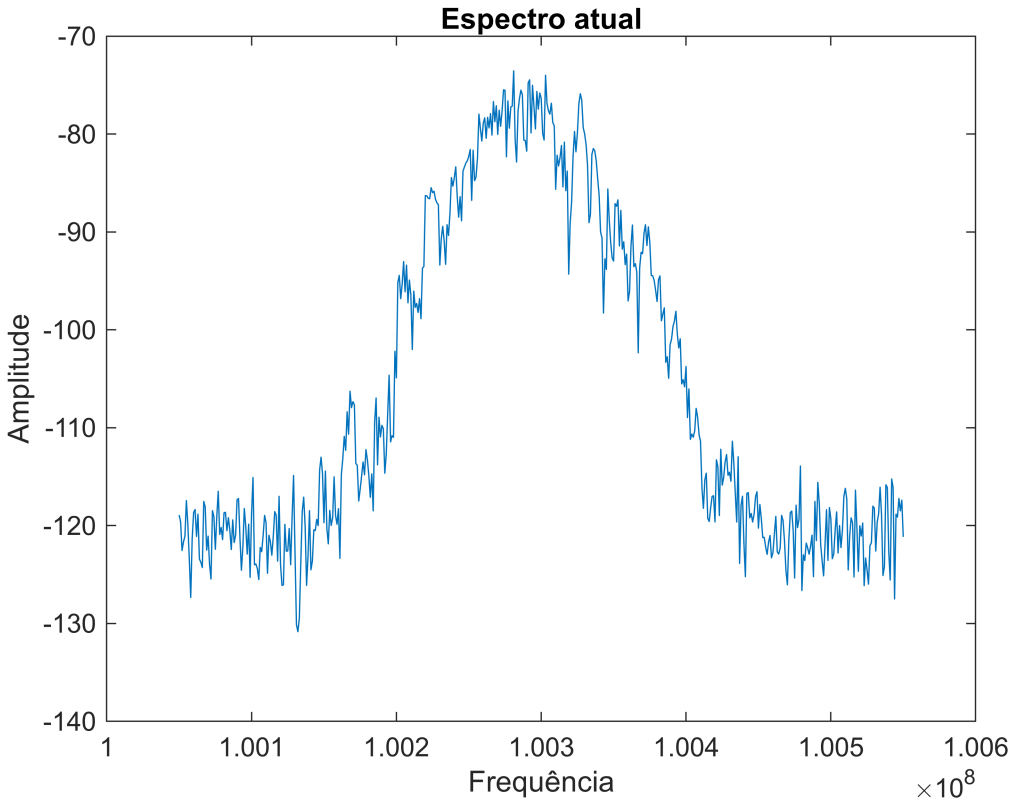
```
    % Só as 10 primeiras linhas
    disp(trace (1:10,:));
```

```
      freq         value
    _________    _______

    1.0005e+08    -118.92
    1.0005e+08     -119.6
    1.0005e+08    -122.54
    1.0005e+08    -121.66
    1.0005e+08    -121.08
    1.0006e+08    -117.44
    1.0006e+08    -119.87
    1.0006e+08    -123.37
    1.0006e+08    -127.34
    1.0006e+08    -121.36
```

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

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




```
legend('off');
```

```
trcs = fcn.getTracesFromUnit(obj, 30);
```

```
ans =
"Trace 1"
ans =
"Trace 2"
ans =
"Trace 3"
ans =
"Trace 4"
ans =
"Trace 5"
ans =
"Trace 6"
ans =
"Trace 7"
ans =
"Trace 8"
ans =
"Trace 9"
ans =
"Trace 10"
ans =
"Trace 11"
ans =
"Trace 12"
ans =
"Trace 13"
```

```
ans =
"Trace 14"
ans =
"Trace 15"
ans =
"Trace 16"
ans =
"Trace 17"
ans =
"Trace 18"
Analyser: Aguardando sincronismo (1/2): ...
:INITiate:IMMediate
ans =
"Trace 19"
ans =
"Trace 20"
ans =
"Trace 21"
ans =
"Trace 22"
ans =
"Trace 23"
ans =
"Trace 24"
ans =
"Trace 25"
ans =
"Trace 26"
ans =
"Trace 27"
ans =
"Trace 28"
Warning: Tektronixs: Trace head contém NaN
ans =
"Trace 29"
ans =
"Trace 30"
```

```
    % Verificação adicional.
    if isnan(trcs)
        warning('Trace com NaN')
    end

    disp("Total size:")
```

```
Total size:
```

```
    size(trcs)
```

```
ans = 1×2
    30   501
```

```
    disp(trcs)
```

```
 -127.5814 -131.8170 -129.0574 -120.6900 -117.1486 -122.4003 -119.9069 -123.1305 -124.5011 -122.7038 -126.7062 -123
 -121.1782 -116.4252 -119.3163 -119.2697 -117.4410 -123.9567 -119.6464 -120.1469 -126.0834 -121.0242 -119.9127 -120
 -131.3005 -118.1023 -116.7847 -116.4160 -117.4212 -120.0097 -117.1708 -118.5767 -115.9676 -116.9352 -120.0461 -119
 -120.4595 -118.2728 -118.2284 -120.3432 -117.5748 -117.5139 -117.9314 -116.3798 -119.3680 -115.8411 -119.3096 -117
 -124.6297 -120.1829 -130.2307 -125.8856 -126.4575 -123.6541 -126.7486 -120.3179 -121.0807 -123.0893 -120.5768 -121
 -122.3747 -122.6565 -122.8359 -118.9386 -117.5675 -123.4268 -123.4034 -123.0279 -116.5615 -118.6091 -118.3483 -117
```

```
 -119.5739 -119.8710 -120.9473 -127.0646 -126.3760 -118.2466 -125.0643 -118.0740 -120.6145 -118.1339 -119.9857 -123
 -128.2187 -123.8889 -119.2512 -117.9343 -117.0522 -117.8921 -117.6668 -123.0607 -118.3553 -118.4966 -120.2272 -124
 -117.2786 -118.3080 -118.1684 -118.9848 -121.1059 -118.7687 -121.4813 -122.0167 -124.5257 -117.2815 -118.7084 -116
 -118.7785 -117.4047 -122.8404 -114.7284 -117.5991 -120.8316 -117.1046 -120.3474 -119.3336 -118.6125 -121.3839 -125
 -124.8006 -113.6496 -115.9969 -122.7110 -121.4571 -120.6714 -119.7826 -114.1513 -113.8095 -114.0373 -115.7469 -128
 -122.6827 -125.1927 -125.2986 -122.0815 -119.9222 -120.7455 -120.2845 -122.5781 -122.8752 -122.4968 -125.2286 -124
```

```
    % % 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: -70.028214
```

```
    if isnan(nivel)
        warning('Nível não retornado.');
    end

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
```