μDX®

Série 100

CONTROLADOR PROGRAMÁVEL

Manual de Utilização

μDX 100 & μDX101

REVISÃO 1.12

Jun/2023

DEXTER Indústria e Comércio de Equipamentos Eletrônicos Ltda.

Av. Pernambuco, 1328 Cjs.307/309 - Porto Alegre - RS - Fones: (51) 3208-0533, 99963-0370

Página Internet: www.dexter.ind.br - E-mail: dexter@dexter.ind.br

µDX 100 & µDX101

Este manual não foi revisado para se adequar a nova norma ortográfica, e nem existe previsão para tal. Com tantas questões relevantes neste País (como o desmatamento na Amazônia, que parece só será resolvido quando a última árvore tombar) há preocupação em modificar a língua pátria, sob justificativas claramente discutíveis. Portanto, espero que o leitor entenda e até aprecie os possíveis tremas que venha a encontrar. Variáveis usadas no programa PG não podem ser acentuadas (de forma similar a endereços de e-mail) e, por isso, os acentos foram suprimidos.

Publicação

DEXTER Indústria e Comércio de Equipamentos Eletrônicos Ltda

Revisão

1.12

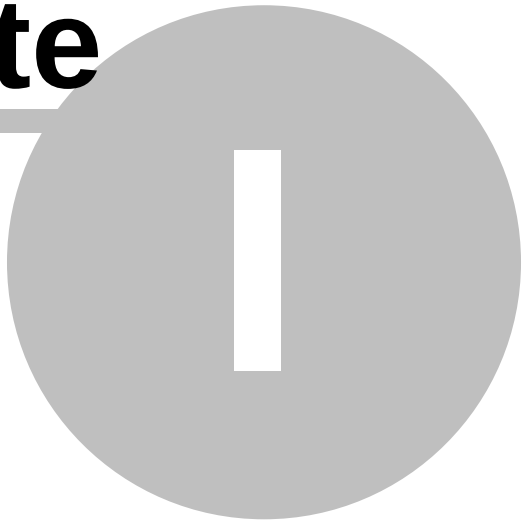
Contato

Claudio Richter

dexter@dexter.ind.br

Conteúdo

Parte



Controlador µDX100

O CONTROLADOR PROGRAMÁVEL µDX100 (pronuncia-se micro-D-X) é um dispositivo eletrônico destinado a controlar processos de automação ou supervisão.

Possui quatro entradas que permitem detectar a ocorrência de determinados eventos como a abertura de portas ou janelas, a presença de luz ou ainda perceber a energização de equipamentos elétricos como um ar-condicionado, lâmpadas ou motores.

As entradas permitem alimentar sensores com 12 V, dispensando fontes adicionais. Assim, é possível conectar diretamente ao µDX100 sensores de proximidade magnéticos ou capacitivos, por exemplo. Também chaves mecânicas de fim de curso, muito comuns em aplicações industriais, são ligadas diretamente ao µDX100.

Cada entrada está relacionada a um LED (indicador luminoso) que liga quando o sinal de entrada é considerado ativo. As entradas são denominadas de E1, E2, E3 e E4.

O controlador µDX100 possui também quatro saídas tipo relé que, atuando como interruptores elétricos, permitem ligar ou desligar aparelhos ou circuitos em corrente contínua ou alternada (DC/AC). Estas saídas podem ligar lâmpadas, motores, eletrodomésticos ou mesmo computadores.

Cada saída também está relacionada com um LED que liga indicando o acionamento do respectivo relé. As saídas são denominadas de S1, S2, S3 e S4.

Para que o µDX100 possa exercer a função que queremos destinar-lhe ele possui, internamente, um microcontrolador para executar as tarefas e uma memória para armazenar o programa.

Este programa que instrui o µDX100 sobre como tratar cada entrada e cada saída fica guardado numa memória que não perde seu conteúdo mesmo quando a bateria interna ou a energia externa não estiverem presentes. Sua construção permite guardar todo o programa por até 40 anos, sem precisar de qualquer fonte de energia.

O programa memorizado possui um ou mais comandos que ordenarão, um de cada vez, o que o µDX100 deve fazer. Depois que todos os comandos do programa forem executados pelo µDX100 ele voltará a fazer tudo novamente, recomeçando desde o primeiro. Os comandos podem ser de vários tipos: chaves comutadoras, temporizadores, operações aritméticas e lógicas, relógios, entre outros.

Para criar esta Programação utiliza-se uma ferramenta de "software" com recursos gráficos que permite montar programas intuitivamente, compondo-se, na tela de um computador, figuras interligadas que representarão os comandos que o µDX100 deverá executar. Esta ferramenta é chamada Programador Gráfico (PG Editor 100).

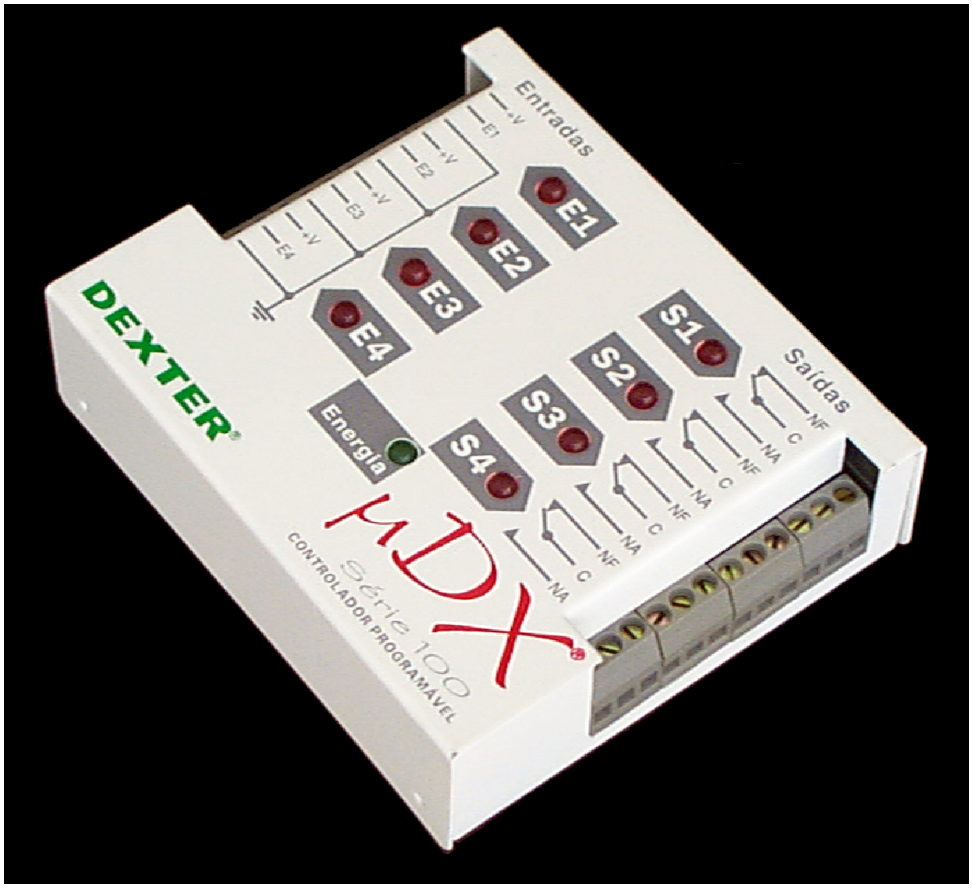
O projeto do µDX100 levou em consideração ainda outras capacitações como: detecção de falta de energia, operação contínua em bateria (com os relés e LEDs desligados) por um período de até 15 dias sem provimento de energia externa e um conector de expansão para entrada e saída de valores com 8 bits, permitindo ter mais oito pontos de entrada e oito de saída (no caso de µDX100 Plus pode-se agregar até 32 entradas e 32 saídas adicionais).

Finalmente, a comunicação em rede local também está presente no µDX100. Através da Rede Local DXNET (especificações desenvolvidas na DEXTER) é possível interconectar até 15 dispositivos que podem trocar informações em alta velocidade. Para melhor aproveitar o potencial da rede DXNET o µDX100 possui uma instrução multi-modo que o habilita a interferir no trabalho de outros µDX100 conectados na rede. Efetivamente isto significa um aumento do número de pontos de E/S e capacidade para elaboração de programas mais complexos que permitam alterar o processo caso algum dos µDX100 na rede sofra avarias ou detecte condições especiais.

Assim, o µDX100 pode reunir estas características:

- Muito baixo consumo: menor que 3 Watts.
- Operação em rede local: até 15 dispositivos independentes.
- Memória de programa não volátil, independente de bateria.
- Relógio de tempo real.
- Operação sustentada por bateria para quando falta energia.

- Programação intuitiva, através de ferramentas gráficas.
- 4 entradas AC/DC de -48 a 48 Volts (expansível à 12 entradas) (expansível a 36 entradas no caso de µDX100+).
- 4 saídas tipo relé com contatos para 10 Ampéres (expansível a 12 saídas) (expansível a 36 saídas no caso do µDX100+).
- 3 entradas analógicas por leitura de largura de pulso (PWM).
- Conector de expansão para mais 16 pontos de entrada/saída (ou mais 48 pontos adicionais no caso do µDX100+).
- Ciclo de execução de programa de 62,5ms/31,25ms/15,6ms ou 3,9 ms.



ATENÇÃO: O controlador µDX100 foi retirado de fabricação. Aconselhamos o uso de µDX101, que possui muito mais recursos e permite migrar diretamente os programas aplicativos.

Funcionamento

O µDX100 possui uma memória não volátil (que não perde o conteúdo mesmo quando falta energia) onde são armazenadas as instruções que lhe dizem o que deve ser feito a cada instante.

Para o µDX100 cada instrução é chamada de bloco de instrução ou de bloco funcional. A memória do µDX100 pode conter até 127 blocos de instrução. No caso do µDX100+, este número aumenta para 256 blocos.

O µDX100 lê e interpreta cada um destes blocos, na seqüência em que foram armazenados, e

chama-se de ciclo a leitura e interpretação completa de todos os blocos memorizados.

Como para qualquer controle ou automatização é necessário o maior grau de paralelismo possível (em qualquer processo sempre pode ocorrer mais de um evento diferente ao mesmo tempo) foi empregado no µDX100 um método de execução de programa chamado Paralelismo Lógico.

Neste método os parâmetros de entrada (estado de ligações e valores de variáveis) são mantidos numa tabela acessível por qualquer um dos blocos de instrução que esteja sendo interpretado. Uma segunda tabela, com os resultados produzidos pela interpretação de cada bloco, vai sendo montada a medida que os blocos vão sendo lidos e interpretados. Assim, cada bloco poderá utilizar qualquer um dos parâmetros de entrada sem que estes sejam alterados devido à interpretação de algum outro bloco. Depois, no final do ciclo, a tabela de saída (com os resultados) é movida diretamente para a tabela de entrada para que os novos valores estejam disponíveis igualmente para todos os blocos no próximo ciclo.

É fácil perceber que esta forma de funcionamento faz com que todos os blocos sejam interpretados em paralelo, o que permite a elaboração de programas segmentados, onde cada parte pode controlar um processo independentemente e ao mesmo tempo que as demais.

Este paralelismo, operado em ciclos, faz com que a atualização da saída de um bloco de instrução para a entrada de um ou mais blocos demore o equivalente ao tempo de um ciclo. Esta demora, ou atraso, deve ser considerado no planejamento de um programa pois a conexão "encadeada" de, por exemplo, 10 blocos de instrução terá uma atraso de 10 ciclos desde o estímulo na entrada do primeiro bloco até a saída no último. Com um tempo de ciclo de 1/16s isto resultaria em um atraso de 1,6 segundos.

A partir da versão de firmware 3.8 o µDX100 realiza um ou mais ciclos extras de execução especial em alta velocidade dentro do tempo de um ciclo normal. Esta execução especial serve apenas para as instruções de chaves interruptoras, acionando a saída destas mais rapidamente e, portanto, reduzindo o atraso total. Note-se que esta aceleração somente ocorre quando a saída da chave será ligada. Devido a lógica de operação do firmware esta aceleração não tem efeito ao desligar-se uma saída de chave.

Além disso, em velocidades de ciclo muito rápidas (como 1/256 s) e programas longos, pode ocorrer que o atraso seja maior que o valor de ciclo multiplicado pelo número de blocos encadeados. Isto porque a execução do programa pode levar mais tempo que um ciclo de execução.

Por limitações do "hardware" interno do microcontrolador empregado, o µDX100 não coloca as variáveis (dados de 8 bits) nas tabelas de entrada e de saída. Assim quando um bloco altera o valor de uma variável este novo valor será empregado por algum outro bloco que utilize a mesma variável ainda no mesmo ciclo. Por exemplo, se um bloco altera o valor da variável V10 para 150 e outro bloco a ser interpretado a seguir (ainda no mesmo ciclo) testa se V10 é maior que 120 o resultado do teste será positivo ainda no mesmo ciclo, como se a Programação fosse seqüencial.

As tabelas de entrada e de saída, no µDX100, contém exclusivamente o estado de cada conexão entre os blocos de instrução: ligado ou desligado. Estas Conexões são também chamadas de NODOS.

A duração de cada ciclo determina a velocidade e a resolução de medida de tempo que o µDX100 pode trabalhar. Existem quatro durações programáveis: 1/16s, 1/32s,1/64s e 1/256s(62.5ms, 31.25ms, 15.625ms e 3,90625ms).

Estas durações são sincronizadas pelo relógio interno mas podem ser dilatadas caso o tempo necessário para a execução de todas as instruções do programa e para as comunicações via DXNET seja maior que a duração escolhida. O tempo total para leitura e interpretação do programa armazenado na memória pode variar conforme o número e tipo das instruções. Ainda, cada comunicação enviada ou recebida via DXNET consome de 10ms até 15ms.

Portanto, a escolha da duração de ciclo pode variar conforme a aplicação do µDX100: processos rápidos e sem comunicação via DXNET podem empregar a duração de 1/64s ou até 1/256s e os processos com muita comunicação podem utilizar a duração de 1/16s.

As entradas (4, digitais) e as saídas (4, tipo relé) do µDX100 são lidas e atualizadas no início de cada ciclo. Além disso elas também são consideradas apenas NODOS, ou ligações, porque

representam apenas um único estado (ligado ou desligado). Para cada entrada existe um LED (indicador luminoso) vermelho para indicar o estado atual dela. Estes LEDs são atuados pelo próprio microcontrolador do µDX100, de forma que seu estado representa exatamente o que o µDX100 está detectando em cada uma das entradas.

As quatro saídas atuam diretamente os respectivos relés, cada um com capacidade de comutar correntes elétricas de até 10 Ampéres. Para cada saída existe um LED que indica se ela está atuada ou não.

Para se programar o µDX100 utiliza-se uma ferramenta fornecida juntamente com ele: o Programador Gráfico (PG Editor 100).

Com o PG é possível programar o µDX100 utilizando-se uma linguagem avançada chamada de PDE : Programação por Diagrama Esquemático. Esta linguagem utiliza recursos gráficos intuitivos para montar os circuitos de controle que irão instruir o µDX100.

Aplicações

A versatilidade e a capacidade do µDX100 permitem imaginar aplicações desde as mais simples até as mais complexas, que podem envolver a intercomunicação de vários µDX100 e utilização de programação distribuída.

O µDX100 pode ser empregado em minuterias inteligentes para edifícios, controle de irrigação de jardins, automatização para paisagismo como o controle de chafarizes, iluminação, ventilação e outros efeitos especiais. Pode operar um semáforo rodoviário completo, inclusive com recursos especiais como onda de verde e horários com temporização diferenciada.

Pode servir como simulador de presença e alarme residencial, controlar o funcionamento do ar-condicionado mantendo o ambiente com temperatura controlada. Pode medir o consumo de energia elétrica de eletrodomésticos, servir como temporizador para fotografia ou mesmo para lembrar o horário para se tomar medicamentos. Até o horário permitido para ligar a televisão ou abrir a geladeira pode ser monitorado e controlado pelo µDX100!

Liga e desliga computadores em horários ou condições específicas. Permite posicionar antenas direcionais, animar festas (luzes e sons de efeito especial) e até mesmo fazer um pisca-pisca para árvore de natal.

Industrialmente ele pode ser empregado para controlar pequenas câmaras de teste por ciclo térmico ("burn-in"), inclusive submetendo o produto em teste a provas de energização intermitente ou verificações de estados progressivos. Pode controlar o uso de máquinas e reduzir o consumo de energia elétrica minimizando o tempo ocioso de máquinas ligadas ou iluminação integral em áreas sem presença de pessoas.

Para o "hobby" o µDX100 pode controlar linhas férreas e até o andamento de modelos de trens, direcionar modelos de embarcações ou mesmo controlar um braço mecânico de robô.

Empregando a Rede Local DXNET é possível automatizar completamente uma residência ou até um parque, quando as entradas e saídas podem ser virtualmente expandidas para 360 ao todo, considerando-se que sejam utilizados 15 µDX100 e todos eles com expansão de mais 8 entradas e 8 saídas - 15x(12+12)=360. No caso do µDX100+ (Plus) este valor pode ser extendido a 1080 entradas/saídas – 15x(36+36)=1080.

Rede Local DXNET

A DEXTER desenvolveu um meio de intercomunicação entre diversos µDX100 ou periféricos para µDX100 que precisem trocar informações para efetuar algum controle de processo: a Rede Local DXNET.

Funciona em topologia tipo barramento, isto é, todos os dispositivos ligados à rede recebem qualquer comunicado simultaneamente.

O modo de operação é Multi-Mestre , ou seja, qualquer um dos dispositivos pode iniciar um comunicado com qualquer outro independentemente de sinalizações (nas redes Mestre-Escravo é

sempre um único dispositivo que sinaliza quando algum outro poderá realizar uma comunicação). Este modo é o mais recomendado para os controles de processos já que oferece equilíbrio de prioridade a todos os dispositivos e permite implementar sistemas mais imunes a falhas.

Eletricamente a rede DXNET é formada por um cabo blindado com uma malha conectada ao comum (0v) e um fio central que conduz as informações. Opera com níveis de 0 e 5 Volts e com velocidade de aproximadamente 9.000 bits por segundo. Esta velocidade é suficiente para permitir conexões a boas distâncias e garantir a troca de dados a uma taxa adequada à própria duração de ciclo dos µDX100 nela conectados.

Se você tiver mais de um µDX eles poderão se comunicar através da Rede Local DXNET. Utilizando o bloco de instrução DXNET o programa elaborado para um µDX pode alterar o conteúdo de uma variável de outro qualquer ou fazer com que seja ligada uma saída (ou mesmo forçar um nodo de entrada como se ela estivesse ligada).

Estes recursos permitem, entre outras coisas, que cada µDX100 conectado à rede processe uma parte de um programa mais complexo, tornando possível controlar grandes ambientes, como fazer uma casa inteira inteligente, buscando conforto e economia, ou automatizando um processo industrial complexo.

Características da rede DXNET

A rede DXNET foi desenvolvida para ser de baixo custo de instalação, empregando-se um simples cabinho blindado (com um fio mais malha) para fazer a interconexão. Este cabinho deve ser pouco capacitivo.

Em cada µDX existe um resistor de 1,8 Kohms ligado entre a linha de comunicação e o +5V. Por isso deve-se limitar em 15 o número de µDX conectados à rede, não só devido ao endereçamento, limitado em 15, mas por que estes resistores ficariam conectados em paralelo, aumentando muito a corrente que cada µDX precisaria drenar para comutar a linha de 1 para 0.

A distância, testada em laboratório, entre uma ponta e a outra da rede é de 100 metros. Este valor pode ser excedido porém sem garantia que o funcionamento seja constante. Ainda assim uma distância muito maior poderia causar danos ao circuito de comunicação, uma vez que a capacitância da linha aumentaria muito e conseqüentemente a corrente de drenagem seria maior.

No µDX existem dois conectores tipo P2, fêmea, que estão interconectados em paralelo. É uma solução de engenharia para facilitar conexão com a rede. Deste modo pode-se interligar os µDX utilizando um cabinho para conectar cada dois µDX, sem precisar de emendas.

Operação da rede DXNET

A comunicação na rede DXNET segue os seguintes princípios:

- O modo de operação é Multi-Mestre, isto é, qualquer um dos dispositivos conectados na rede pode iniciar uma comunicação.
- Todos os µDX100 estão sempre atentos à rede e "ouvem" a linha reconhecendo o cabeçalho da comunicação, onde existe um caracter de início e um segundo com o comando e o endereço de destino. Se o endereço de destino for diferente do endereço próprio do µDX a rotina de recepção é interrompida e o µDX continua o serviço que estava fazendo. A excessão é o endereço 0 (zero) que todos os µDX reconhecem, sempre.
- Toda comunicação inicia com um bit de start (sinal em nível zero com duração de 2ms) e somente termina após o µDX de destino haver respondido ao comando ou que tenha ocorrido um "Time-Out". Este último ocorre se nenhum µDX na rede possui o mesmo endereço indicado no comunicado.
- Antes de qualquer dispositivo na rede iniciar uma comunicação ele deve verificar se a linha está livre por um período mínimo que varia conforme seu endereço na rede e a qual a tentativa de comunicação atual.
- O tempo de comunicação (incluindo-se o tempo gasto nas tentativas) consome, gradativamente, o tempo livre para comunicações por ciclo. Se este tempo livre for excedido a

duração do ciclo será dilatada em função do excesso, podendo causar erros nas temporizações de frações de segundo.

- Ao programar um µDX100 conectado junto com outros na rede DXNET tenha certeza de que ele responde a um endereço diferente que os demais para que somente ele receba o programa. Com todo µDX sai de fábrica com endereço 0 (zero) deve-se programar cada um individualmente, com a conexão DXNET apenas entre o computador e o µDX. Depois disso, tendo ele recebido um endereço próprio, qualquer nova programação já poderá ser feita em rede.
- O endereço de cada µDX100 fica armazenado em memória não volátil, completamente independente das pilhas.

Programação usando a rede DXNET

O uso do bloco de instrução DXNET é muito simples.

Deve-se ter em mente que os blocos DXNET apenas transmitem. Assim para ler o conteúdo de uma variável em um certo µDX é preciso que o programa nele tenha um bloco DXNET que transmita o valor desta variável para o µDX local.

Apesar de parecer uma limitação, na maior parte dos processos controlados é preciso apenas transmitir informações quando ocorrem condições especiais. Para os outros casos basta elaborar os programas dos diversos µDX de acordo com um planejamento adequado.

É recomendável restringir o número de comunicados ao mínimo necessário, evitando principalmente os comunicados simultâneos. Programe o disparo da comunicação conforme o tempo de ciclo do processo controlado. Ele é determinado pelo tempo que o processo controlado precisa para modificar-se de forma significativa. Por exemplo o controle de temperatura de uma sala é bastante lento pois a mudança de um grau centígrado pode demorar desde dezenas de segundos até muitos minutos. Logo, informar a temperatura da sala a um outro µDX pode ser feito a cada minuto sem risco de perda de informação.

O nodo de entrada do bloco DXNET deve ser mantido em cada estado pelo máximo de tempo possível. No caso de enviar o valor de uma variável, quanto mais tempo o nodo de entrada ficar em 1 mais tempo haverá para retentativas caso haja algum erro de comunicação. No caso de enviar estado de nodo o tempo para retentativas vai ser conforme a duração que tiver o novo estado do nodo de entrada.

> **Nota:** No caso de µDX+ é possível expandir a rede DXNET a 256 endereços usando a informação adicional de CONJUNTO. Como cada um dos 16 conjuntos DXNET é formado de 16 endereços DXNET o total de dispositivos endereçaveis chega a 256. Entretanto, por limitações de corrente (como já comentado) esta opção só é operacional no caso de interligar os vários conjuntos via rede RS485 ou Rádio-Transmissão (com o módulo de Modem para µDX).

Instalação e Troca de Pilhas

Internamente ao µDX100 existe um suporte para 4 pilhas do tipo AA (pilhas pequenas). Estas pilhas irão suprir de energia o microcontrolador do µDX100 quando ocorrer interrupção no suprimento através da Fonte de Alimentação (falta de energia na rede elétrica ou desligamento da Fonte de Alimentação).

Com este suprimento de energia auxiliar o µDX100 mantém o funcionamento do relógio interno e também continua a execução completa do programa que estiver em sua memória. Apenas os relés, os LEDs e algum eventual circuito de expansão é que não recebem força desta energia auxiliar fornecida pelas pilhas.

Desta forma o consumo fica em torno de 2mA, o que permite o processamento contínuo (inclusive com comunicação via DXNET) por até 15 dias (empregando pilhas alcalinas novas).

Quando a energia suprida pela Fonte de Alimentação está presente as pilhas são eletronicamente desligadas, mantendo suas cargas preservadas.

> **ATENÇÃO:** O µDX100 não possui circuito para reconhecimento de quando as pilhas estiverem sem carga. Exemplificando, se o µDX100 precisar funcionar apenas com as pilhas por cerca de 1 hora por dia (mantendo a Fonte de Alimentação energizada durante as outras 23 horas) as pilhas somente precisarão ser substituídas uma única vez por ano.

Para substituir as pilhas (ou instalá-las a primeira vez) é preciso abrir a caixa do µDX100.

Desligue os conectores da Fonte de Alimentação, da Rede Local DXNET e da expansão (se tiver). Abra a caixa do µDX100 forçando levemente as laterais para afastarem-se dos encaixes que prendem a tampa ao fundo.

Puxe a tampa cuidadosamente para cima. Cuidado com os LEDs que estão montados presos a pequenos conectores de dois contatos. Caso algum dos LEDs saia do lugar observe a posição dos LEDs adjacentes para saber como encaixar o que saiu (ao contrário de uma lâmpada, o LED tem polaridade e não liga se for invertido).

Caso já existam pilhas (que neste momento estão mantendo o µDX100 em operação) volte a ligar o conector da Fonte de Alimentação. Assim o µDX100 não vai parar (e o relógio será mantido em funcionamento) quando as pilhas forem retiradas.

Retire o suporte das pilhas do encaixe, tomando o cuidado para não puxar em demasia e forçar os fios de ligação.

Coloque 4 pilhas novas (com carga completa) do mesmo tipo e observando a posição de cada uma devido a polaridade (a mola do suporte de pilhas é sempre ligada ao terminal negativo das pilhas).

> **ATENÇÃO:** Nunca utilize pilhas de tipos diferentes ao mesmo tempo, como uma mistura entre pilhas convencionais e alcalinas. Cada tipo possui um modo de funcionamento próprio que não permite combinação adequada com outros modelos.

Depois de instaladas as pilhas no suporte volte a encaixá-lo na fixação deste na parte de baixo da caixa. Desligue o conector da Fonte de Alimentação (se estiver ligado).

Verifique se os LEDs estão bem alinhados entre si antes de aproximar a tampa para o fechamento. Encaixe a tampa alinhando os furos superiores com os LEDs. As quatro pequenas depressões da tampa irão fixar-se nos furos da parte de baixo da caixa feitos para este fim.

Volte a ligar os conectores da Fonte de Alimentação, DXNET e expansão (se estiver utilizando).

Note que, uma vez instaladas as pilhas, sempre que não houver energia elétrica sendo suprida pelo conector de ENERGIA as pilhas estarão sendo gastas. Assim, não basta parar o programa instalado no µDX100 para evitar o consumo das pilhas.

> Nunca armazene o µDX100 com pilhas instaladas. Caso ele não vá ser usado por algum tempo retire as pilhas para evitar que descarreguem.

Pilhas Recarregáveis

É possível instalar pilhas recarregáveis no µDX100, evitando o incômodo da troca periódica destas em aplicações que necessitem do relógio de tempo real. Podem ser instaladas 4 pilhas recarregáveis tipo AA, de 500mAh ou 750 mAh. Para que as pilhas sejam recarregadas quando o µDX100 estiver com alimentação elétrica, basta fechar a conexão ("jumper") JP14, existente na placa de circuito impresso (esta conexão vem aberta de fábrica, prevendo pilhas comuns)(Esta conexão só está presente em placas versão 1.4 ou maior).

Nunca feche esta conexão ("jumper") com pilhas comuns (não recarregáveis) instaladas no µDX100. Pode ocasionar vazamentos de ácido das pilhas.

ATENÇÃO: No caso de pilhas comuns, a tensão de alimentação do µDX100 (Vcc) atinge 6V (4x1,5V). Isto não provoca nenhum problema no circuito do µDX100. Entretanto, tal fato deve ser considerado ao projetar-se circuitos a serem ligados ao conector de expansão. Já no caso de pilhas recarregáveis, a tensão fica em 4,8V (4x1,2V); mais próximo dos usuais 5V (quando alimentado pela rede elétrica a tensão no µDX100 - Vcc - é de 5V).

Entradas e Saídas

As Entradas

As quatro entradas digitais do µDX100 estão localizadas, todas, numa única lateral da caixa (lado esquerdo quando a caixa é vista de frente).

Elas estão codificadas como E1, E2, E3 e E4. Cada uma utilizando três contatos: um para o +V, um para a entrada (E1 até E4) e um para o 0V (conhecido também como "GND", massa, retorno ou comum); conforme representado pela serigrafia da tampa.

O +V é uma conexão internamente ligada ao conector da Fonte de alimentação (Energia) e permite que algum circuito externo de baixo consumo seja energizado (emprega-se também o contato de 0V). Tal circuito pode servir como um conversor ou adaptador ativo para que seja possível, por exemplo, detectar com uma entrada digital os sinais vindos de sensores de luz, pressão, umidade ou mesmo temperatura.

Além disso o contato de +V é empregado para ativar a entrada quando um interruptor é utilizado como sensor. Neste caso o interruptor fica ligado entre o +V e o contato da entrada. Quando o interruptor fechar (permitir a passagem de corrente elétrica) o circuito da entrada detectará o sinal tornando-a ativa.

A entrada é considerada ativa quando a tensão presente nela estiver entre +2 Volts e +48 Volts (valor positivo medido em relação ao 0V), e é considerada inativa quando a tensão ficar entre -48 Volts e cerca de +0.9 Volt. Esta região de incerteza, entre +0.9 Volt e +2 Volt, é comum para os circuitos digitais convencionais e suficiente para todas as aplicações práticas do µDX100.

CUIDADO: Os µDX com número de série a partir de 03800011 não possuem resistor limitador entre o contato +V e o conector de Energia. Assim algum curto-circuito acidental entre o contato +V e o 0V poderá causar danos ao circuito impresso ou à Fonte de Alimentação.

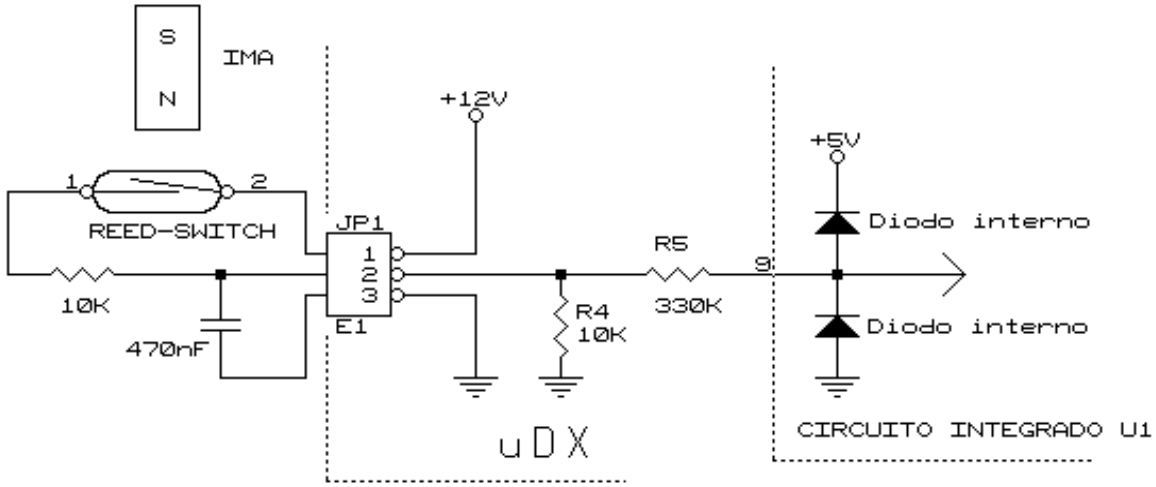
ATENÇÃO: As entradas do µDX100 NÃO SÃO ISOLADAS GALVANICAMENTE, isto é, não é possível ligar a qualquer uma delas algum fio conectado à rede elétrica domiciliar. Se isto for feito poderá causar danos graves no computador que estiver ligado na DXNET e no próprio µDX. Se for preciso detectar a presença de energia elétrica utilize um circuito isolador (tipo OPTO-ACOPLADOR) como sugerido no anexo DIAGRAMAS ESQUEMÁTICOS. A DEXTER comercializa um circuito com 4 opto-acopladores para o µDX100.

Se alguma aplicação em que for empregado o µDX gerar muito ruído elétrico que possa ativar alguma das entradas por interferência, deve-se utilizar um pequeno circuito de filtragem nas entradas sensíveis. Este filtro é formado por um resistor e um capacitor, montados como mostrado nas figuras seguintes.

Alguns exemplos de circuitos de sensoriamento são mostrados a seguir.

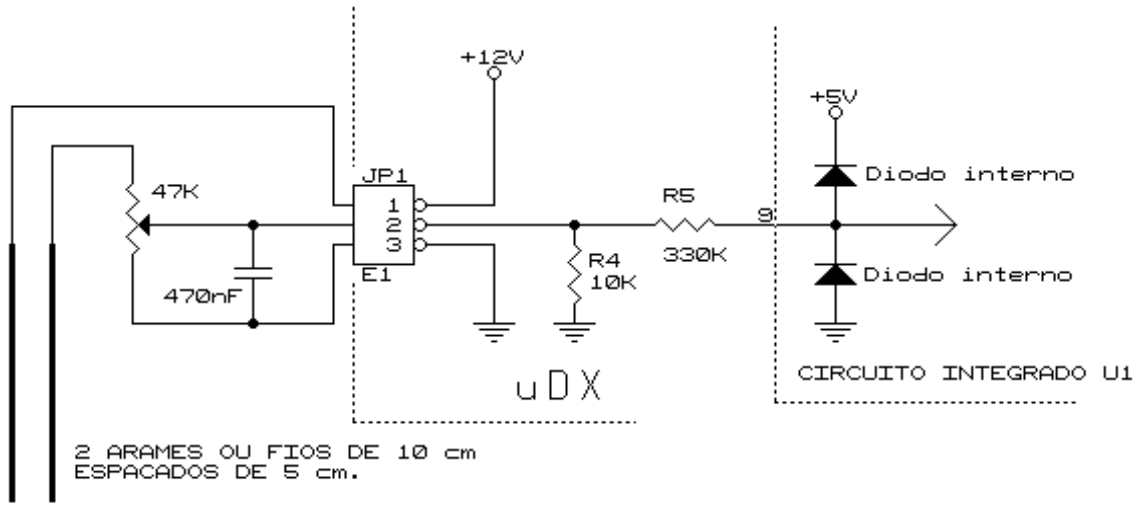
Sensor tipo Reed-Switch

Este tipo de sensor é adequado nas instalações de alarmes, detectando a abertura de portas ou janelas. O esquema mostrado na figura apresenta, também, parte do circuito interno de uma das entradas do µDX (ver Anexo C - Diagramas Esquemáticos). O resistor interno de 10K serve para descarregar o capacitor utilizado como filtro quando o Reed-Switch abre.



Sensor de Umidade

Para que alguma das entradas do µDX detecte a presença de umidade no solo de um gramado ou de vasos (também o transbordamento de piscinas ou um sensor de nível para algum processo industrial) é preciso apenas um circuito externo muito simples. Como a impedância de entrada é cerca de 10K (resistência entre o sinal de entrada e o comum), e considerando que o µDX detecta entrada ativa a partir de 2 Volts, para isto bastará uma resistência menor que 50K (entre o +V e o sinal de entrada).

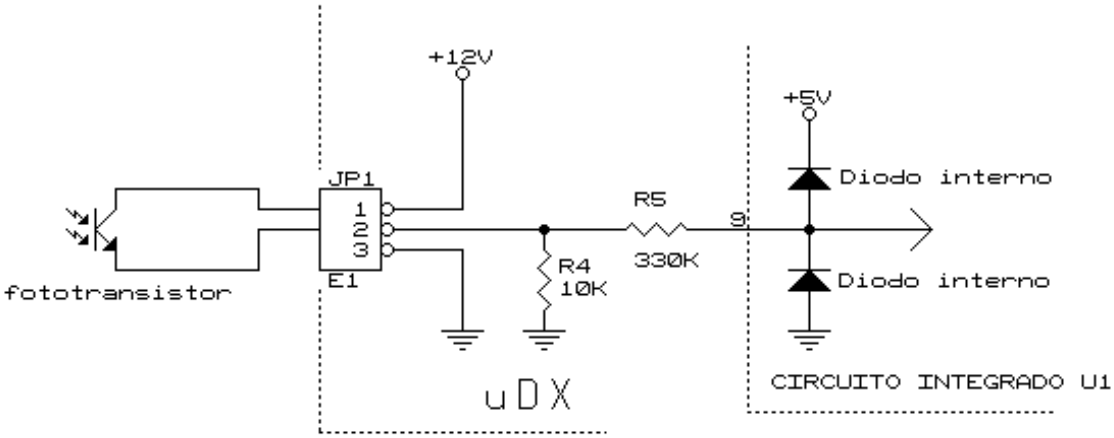


Nas experiências práticas constatou-se que a resistência medida na presença de umidade é bem menor do que isso. Portanto, com um resistor variável (trimpot) montado como o esquema abaixo pode-se ajustar o limiar de sensibilidade do sensor adequado para cada caso.

Sensor de Luz

Com este esquema muito simples pode-se detectar a incidência de luz em um foto-transistor (pode ser o TIL81 ou outro equivalente). Basta conectar o foto-transistor como mostrado. Se for empregada uma barreira ótica (componente com um emissor de luz e um foto-transistor) o emissor de luz poderá ser alimentado a partir do +V presente no conector de entrada do µDX.

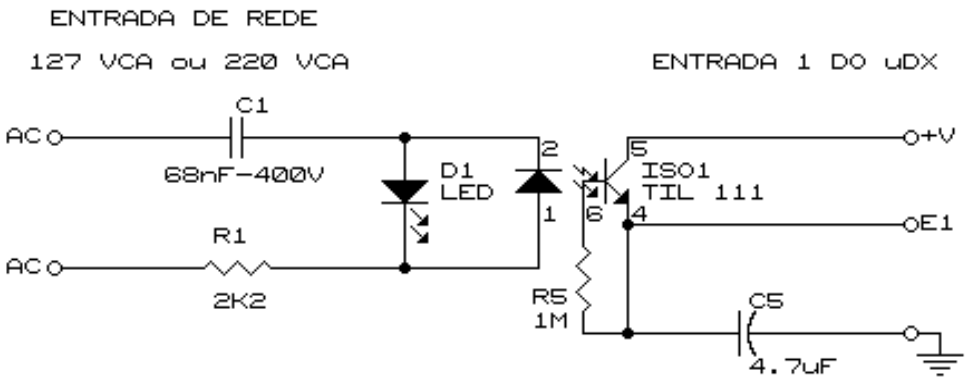
Utilize um resistor de 1K8 em série com o emissor de luz para limitar a corrente ao nível necessário.



Detector de tensão AC Opto-acoplado

Com este circuito pode ser detectada a presença de tensão AC em 110V ou 220V, com a segurança do sinal ficar isolado do circuito do µDX100. Esta isolação é necessária para evitar curto-circuito entre diversas entradas que monitorem sinais AC de alta tensão e também protegem o µDX de eventuais descargas elétricas que possam ocorrer. O LED D1 ilumina-se quando a tensão AC está presente, também protegendo o emissor de luz do opto-acoplador contra a tensão reversa do semi-ciclo não aproveitado por ele.

O capacitor de 4,7µF(C5) age como filtro, com um período maior que o ciclo de rede 60Hz (16,67 ms). Para ligarmos a esta entrada uma tensão de 12Vdc em vez dos 127 ou 220 VAC basta substituir o capacitor C1 por um curto-circuito (e colocar o LED em série com o LED do opto-acoplador, de forma que ambos liguem quando for aplicada tensão na entrada).

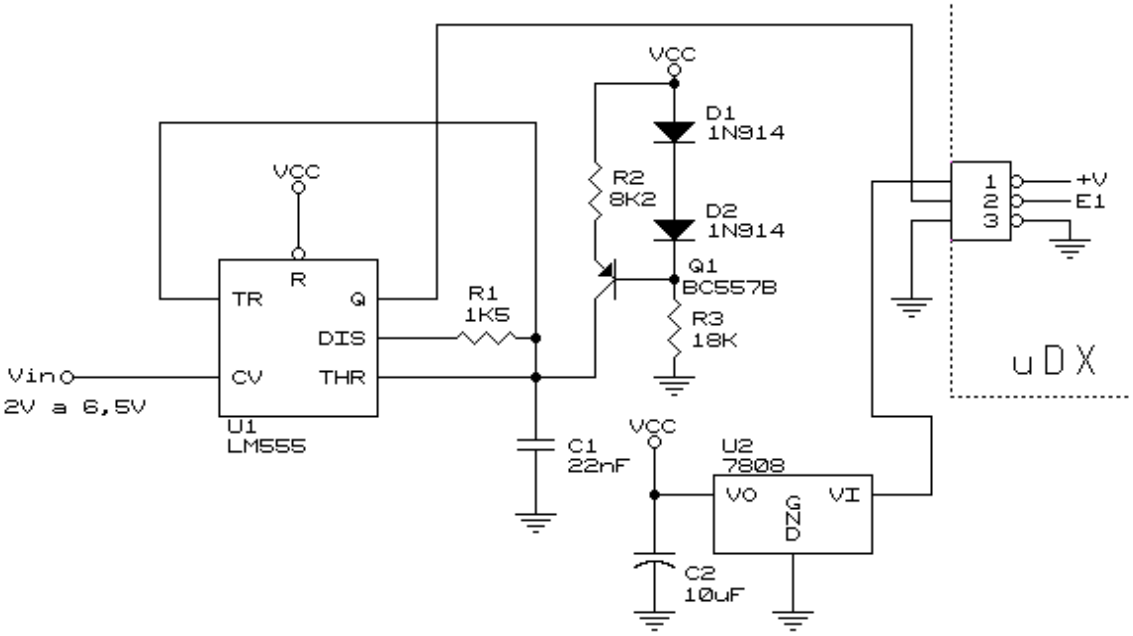


Conversor Tensão / Largura de Pulso (PWM)

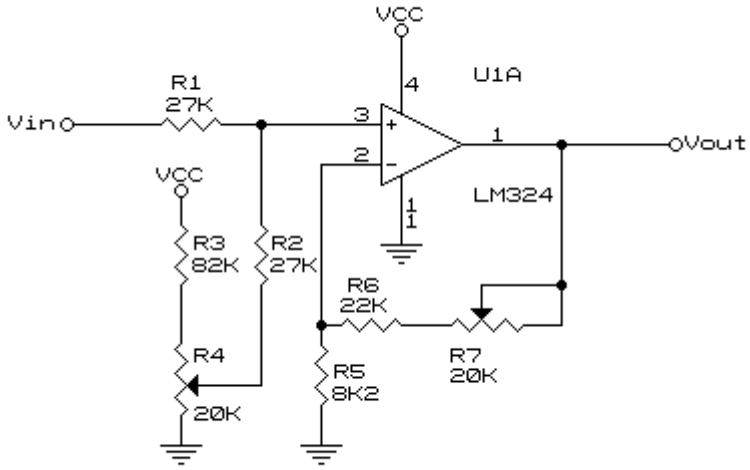
O circuito apresentado faz a conversão de um nível de tensão da entrada para uma largura de pulso proporcional na saída. Utilizando-se o bloco de instrução PWMin, do µDX, pode-se fazer a leitura desta largura de pulso em um valor de 8 bits a ser colocado em uma variável interna (ver FUNÇÃO - PWMin).

Foi utilizada a entrada de ajuste de comparação do 555 para que a posição de gatilhamento, sobre a rampa linearizada pela fonte de corrente, mova-se conforme a tensão aplicada. O regulador de tensão 7808 garante uma tensão de alimentação para o circuito com boa estabilidade, já que a tensão disponível no conector de entrada do µDX (+V) não é regulada.

Utilizando-se os limites indicados na figura (2V a 6,5V) o resultado da função PWMin terá os valores de 50 a 200, aproximadamente. A conversão dentro destes limites é linear.



Como os limites de tensão de entrada estão em uma faixa pouco usual, recomenda-se empregar um circuito adicional de correção da faixa que ajusta os limites de 0 a 1 Volt. Além disso, o circuito da página seguinte, quando conectado à entrada 5 do LM555, permite uma alta impedância de entrada. Com isso, podemos fazer uma entrada analógica de corrente - 4 a 20 mA. Basta colocar um resistor de 47R em paralelo com a entrada e ajustar os trimpots para obter uma leitura correspondente à corrente de entrada no µDX (por exemplo, 40 para 4mA e 200 para 20mA).



Aqui os trimpots R4 e R7 servem, respectivamente, para calibrar o zeramento e o ganho do circuito em função da tolerância dos componentes. Com este circuito pode-se ajustar para que a medida de 0 a 1 Volt corresponda a valores como 80 a 180 obtidos pela função PWMin. Depois bastará subtrair 80 do valor da variável para conseguir uma correspondência de 0 a 1 Volt = 0 a 100.

As Saídas

As quatro saídas do µDX100 são do tipo relé com um contato reversor, isto é, possui um contato central e dois adjacentes, ficando um deles ligado e outro desligado quando a saída está inativa. Quando a saída fica ativa o contato central troca de lado deixando o primeiro contato adjacente desligado e o outro ligado.

Esta particularidade permite que se ligue ou que se desligue algum dispositivo quando a saída for acionada, bastando escolher qual dos contatos adjacentes deva ser empregado.

A serigrafia na tampa da caixa (lado direito da tampa) mostra estas conexões e a representação dos contatos dos relés.

A designação dada às saídas é S1, S2, S3 e S4.

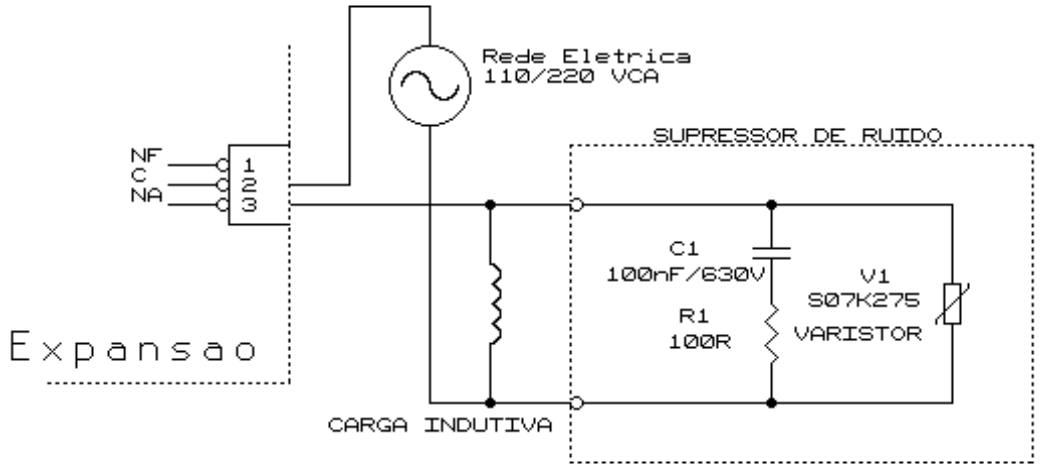
Note que os relés dispõem de isolação galvânica entre a bobina de acionamento e os contatos. Isto torna possível ligar dispositivos à rede elétrica domiciliar para serem atuados diretamente pelos contatos do relé, sem risco algum para o µDX100 ou algum outro equipamento conectado na Rede Local DXNET.

Supressores de Ruído Elétrico

No caso do µDX100 acionar cargas indutivas (contactoras, válvulas pneumáticas, etc) é necessário instalar supressores de ruído junto à carga indutiva (em paralelo com essa). O supressor evita que, ao abrir o relé do µDX ligado à carga indutiva, forme um arco de alta tensão nos contatos. Este arco, embora não danifique permanentemente o controlador programável µDX, diminui muito a vida útil dos relés e, em casos extremos, pode abortar a execução do programa no µDX, sendo necessário reinicializá-lo. A seguir temos o diagrama elétrico do supressor e sua ligação a uma carga. A DEXTER pode fornecer supressores de ruído já montados para a sua aplicação.

Abaixo temos uma lista de precauções contra ruídos elétricos na instalação do controlador programável:

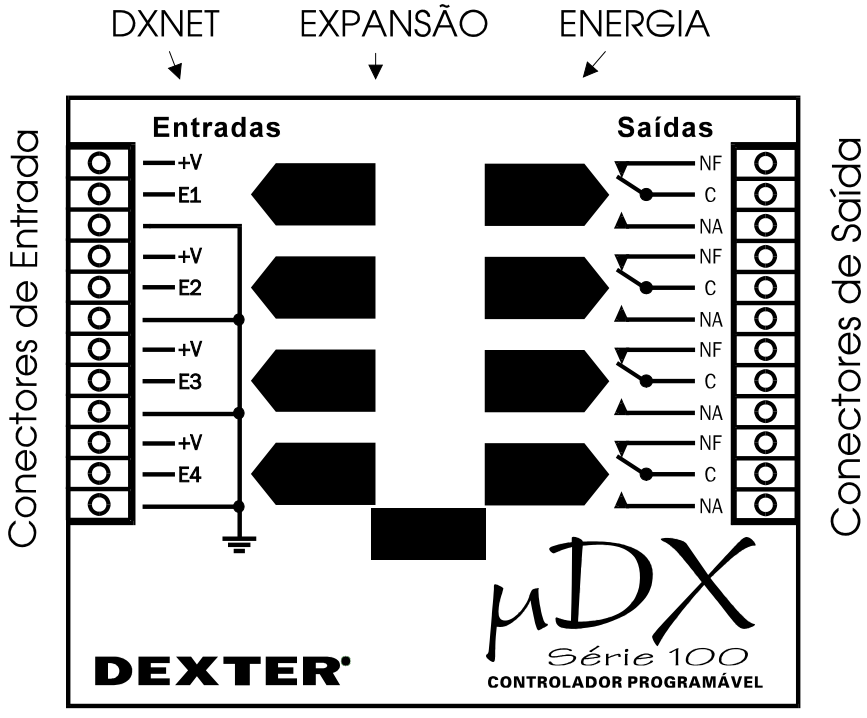
- Instalação de supressores de ruído, como já frisado.
- Separação física dos cabos de sinal (DXNET, alimentação elétrica do µDX, sinais de entrada para µDX) dos cabos de potência (contactoras, motores, etc.). Nos casos em que os cabos devem correr juntos, providenciar um eletroduto metálico (devidamente aterrado) para blindar os cabos de sinal.
- O terra lógico do µDX (disponível no conector das entradas E1 a E4) não deve ser aterrado.
- No caso de ruídos advindos da fonte de alimentação, é necessário providenciar uma linha de alimentação separada para o µDX e outros equipamentos eletrônicos da instalação.



A partir da versão 2.1 de placa impressa do Controlador Programável µDX Série 100 foram inseridos vários componentes adicionais para supressão de transientes elétricos. Isso evita que o transiente perturbe o funcionamento do controlador. Entretanto, no caso de cargas indutivas, ainda é interessante a inserção de supressores de ruído elétrico para aumentar a vida útil dos relés de saída do µDX.

Conectores

Os conectores do µDX100 são os seguintes:



Entradas

São quatro módulos de três terminais cada um, montados na lateral esquerda do µDX100. Como

indicado na serigrafia do painel o primeiro terminal de cada módulo é para o +V. O segundo terminal é para o sinal de entrada e o terceiro é o comum (terra, Zero Volts, "ground"). Pode-se alimentar algum circuito externo utilizando-se o +V e o comum para isso. Todos os +V estão interligados, assim como os comuns.

> **Atenção:** não existe isolação galvânica entre qualquer um dos terminais e o restante do circuito interno do µDX.

Saídas

São quatro módulos de três terminais cada um, montados na lateral direita do µDX100. Como indicado na serigrafia do painel o segundo terminal de cada módulo corresponde ao contato móvel interno de cada relé de saída. Os outros dois terminais referem-se aos contatos dos relés, Normal Fechado e Normal Aberto.

Os relés possuem isolação galvânica entre si e entre os contatos e o circuito interno do µDX.

Não existe interconexão interna entre os relés.

Energia

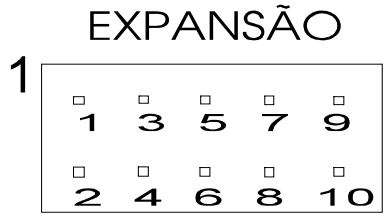
O conector para ligação com a fonte de alimentação fica localizado na lateral traseira do µDX100, ao lado do conector da expansão.

A energia externa para alimentar o µDX100 deve ser entre 9 e 14 VDC suprindo até 250mA.

Como representado no desenho ao lado dele o contato central é ligado ao comum (Zero Volts, "ground"). O contato externo é ligado aos circuitos dos relés, na fonte interna que reduz a tensão para 5 Volts e aos terminais do conector das entradas.

Expansão

É um tipo barra de pinos, com 10 contatos em duas filas de 5. O primeiro fica no canto superior esquerdo, sendo a numeração dos contatos a seguinte:



Serve para conectar ao µDX100 algum circuito externo para a entrada e saída de dados de 8 bits, como a Expansão de Entradas/Saídas.

Significado dos contatos:

1 - Comum (0V)
2 - +V (diretamente ligado à energia)
3 - Pulso de carga ("Load")
4 - +5V (máx. 10mA)
5 - Pulso de relógio ("Clock")
6 - Sem conexão
7 - Entrada de dado serial ("Din")
8 - VCC (+5V garantidos pelas pilhas quando falta energia, máx. 5 mA)
9 - Saída de dado serial ("Dout")
10 - Sem conexão

DXNET

São dois conectores tipo P2, fêmea, que estão internamente ligado em paralelo. Em qualquer um dos dois podem ser conectados os cabos para rede DXNET ou o cabo do adaptador para a comunicação com o computador.

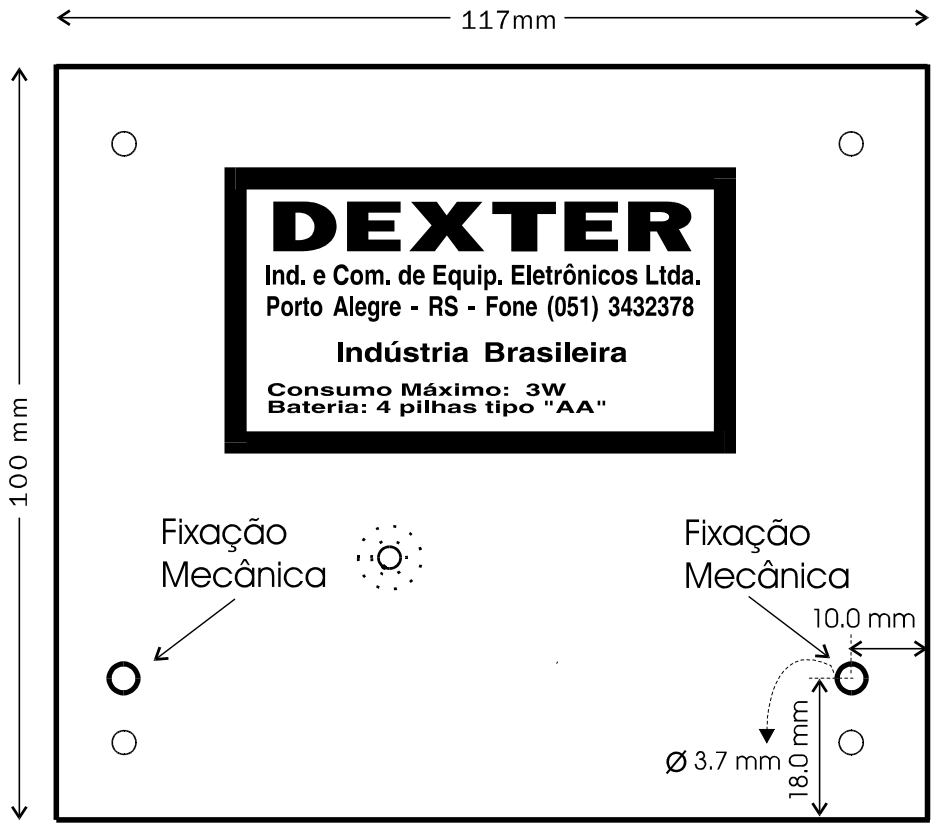
O contato da ponta é o sinal e o contato do corpo é o comum (Zero Volts, "Ground").

O sinal está ligado diretamente ao microcontrolador do µDX100 e a um resistor de 1,8K (que vai ligado ao VCC).

> Cuidado para não aplicar no sinal tensões superiores à +5 Volts, nem inferiores à -0,5 Volts, sob risco de dano irreversível do microcontrolador.

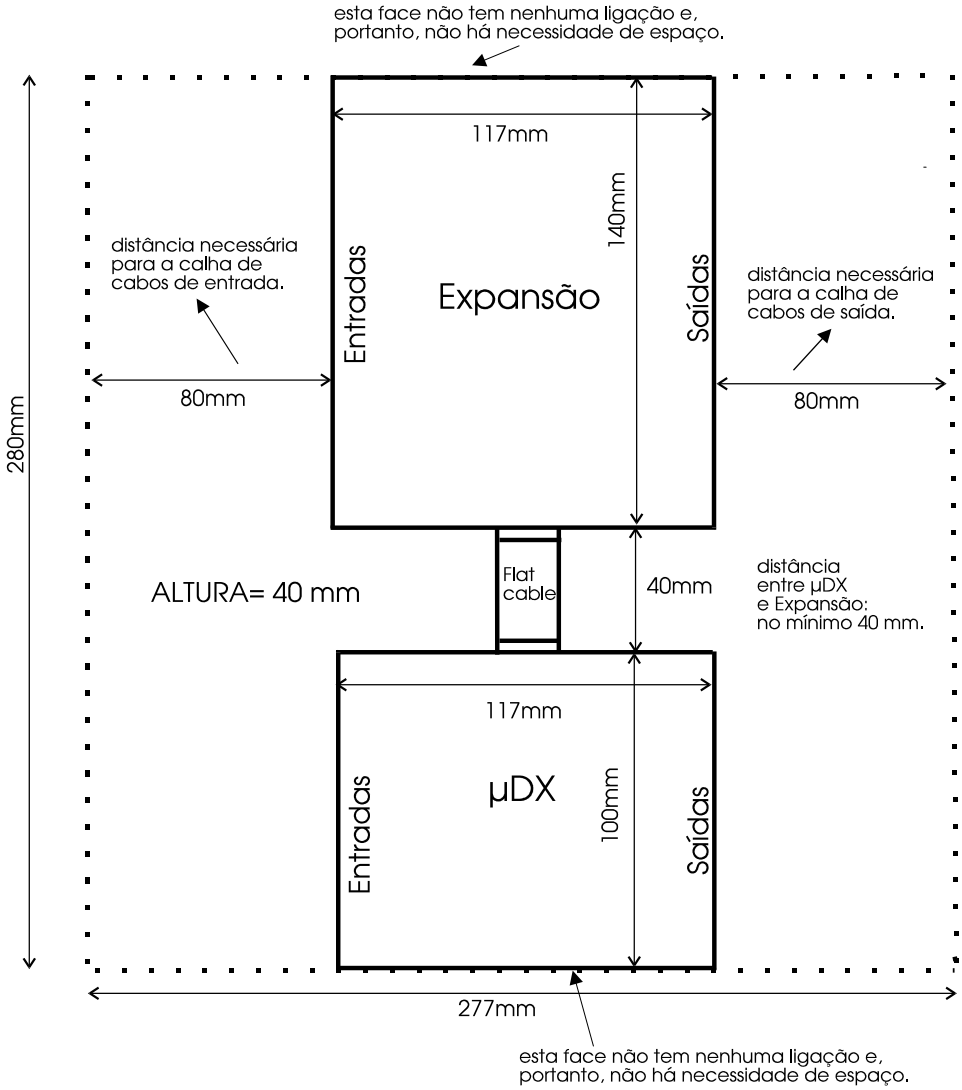
Fixação Mecânica

Para fixar o µDX100 existem dois furos na face inferior da caixa, conforme o desenho abaixo. Estes furos são disponíveis abrindo-se o µDX (retirando a tampa superior) e deslocando para fora de seu suporte o porta-pilhas.



VISTA DA PARTE INFERIOR DA CAIXA

Área ocupada por µDX + Expansão



Especificações Técnicas

Características Gerais

- 52 Instruções, incluindo lógica e aritmética de 8 bits.
- 16 timers de 8 bits (compartilhados com as variáveis) (64 para µDX100+).
- 16 variáveis de 8 bits (64 para µDX100+).
- Estruturação do programa por rede nodal.
- Execução do programa em modo de paralelismo lógico.
- Memória com capacidade para 127 blocos de instruções e 63 nodos (256 blocos e 192 nodos, no caso de µDX100+).
- Quatro durações de ciclo: 1/16, 1/32, 1/64 e 1/256 segundo.
- "Watch-Dog-Timer".
- 4 entradas e 4 saídas digitais, expansível para 12 entradas e 12 saídas (36 entradas e 36 saídas no caso do µDX100+).
- Entrada de contagem rápida até 3000 Hz (apenas para µDX100+).
- 3 entradas analógicas (por PWM).

Características Elétricas

- Alimentação (ENERGIA): 9-14VDC @ 250mA (máximo)
- Bateria Interna: 4 pilhas tipo AA (ou baterias recarregáveis)
- Consumo na alimentação pela bateria: aproximadamente 2mA (5mA no caso de µDX+).
- Oscilador Central: 4,194304 MHz ou 16,777216 MHz (firmware ≥ 6.3)
- Temperatura de operação: 0°C até 60°C.
- Entradas:
    - Lógica normalmente a zero (com "pull-down" de 10K).
    - Detecção de 1 lógico: +2.0 até +48.0V
    - Detecção de 0 lógico: -48.0V até +0.9V.
    - Tensão máxima na entrada (1 minuto): 120VCA.
    - Freqüência de entrada: DC - 10Hz (260KHz para bloco PWMin).
    - Tensão em +V (em aberto): 9-14VDC (conforme ENERGIA)
- Saídas:
    - Tipo de saída: 1 reversor (por cada saída)
    - Tensão nos contatos: 30VDC/220VCA
    - Corrente máxima: 10A
    - Vida útil sem carga: 100.000.000 operações
    - Vida útil com carga resistiva máxima (CA): 80.000 operações
    - Isolação (entre bobinas e contatos): 2.000Vef (1 minuto)
- Conector de Expansão:
    - Saída de alimentação +V: máx. 100mA (9 a 14 VDC)

- Saída de alimentação +5V: máx. 8mA (+V=9VDC); máx. 50mA (+V=14VDC)
- Saída de alimentação VCC: máx. 5mA (consumo direto das pilhas)

Versões de Software

Firmware do µDX100

Trata-se do software interno do controlador µDX100. Sua versão pode ser obtida ao ler o Status do controlador.

V 0.0	22/05/1993	Versão Inicial.
V 0.2	20/06/1993	Aperfeiçoamentos gerais.
V 0.3	05/07/1993	Correções e ajustes (software ainda parcial).
V 0.4	10/07/1993	Funções matemáticas e de comparação.
V 0.5	11/07/1993	Expansão.
V 1.0	09/08/1993	Arrumação geral nos registradores.
V 2.0	19/08/1993	Junção VAR/TIMERS, DXNET, correção de MONO. Depuração dos timers. Sincronização do ciclo pelo relógio. Correções no relógio.
V 2.2	17/09/1993	Função PULSO e ATRASO.
V 2.3	27/11/1993	Novo protocolo de comunicação DXNET.
V 2.4	20/01/1994	Relógio com minutos, reset timers.
V 2.5	22/01/1994	DXNET, nodo 62 a zero.
V 2.6	29/01/1994	DXNET, PULSO, OSCILADOR, outras correções.
V 2.7	08/02/1994	PWM, correções.
V 2.8	08/02/1994	TURBO com contador de tempo separado para timers.
V 3.0	19/02/1994	Bi-estável, correções gerais, DXNET mais rápida.
V 3.2	09/07/1994	Retirado o timeout, correções na comunicação, power-on-reset.
V 3.3	28/08/1994	Retirado checksum, oscilador corrigido.
V 3.4	30/08/1994	DXNET comando 14.
V 3.5	03/09/1994	Lê EEPROM em bloco, propaga chaves mais rápido em ciclo morto.
V 3.6	15/09/1994	Retirada propagação de chaves, faz acesso a EEPROM 93C66.
V 3.7	17/09/1994	Versão 3.6 mais compacta, recolocada propagação de chaves.
V 3.8	19/09/1994	Versão com aprimoramento na velocidade de leitura da EEPROM. Recolocadas instruções de configuração das portas. Corrigida falha de sincronismo nos tics de relógio durante ciclo de chaves.
V 3.9	25/12/1994	Permite gravar/ler EEPROM a qualquer tempo sem perder sincronismo na leitura auto-incrementada.
V 4.0	02/03/1995	Corrigido erro nos blocos POWER-UP e POWER-DOWN. Esta versão não tem a modificação da versão 3.9 devido a perda de performance.
V 4.1	23/10/1995	Inúmeras modificações no código com o intuito de compactá-lo e aumentar a performance de execução do µDX. Implementada a velocidade de execução de 1/256 segundos. Reset da Expansão de I/Os quando µDX é resetado. Variável do PWM assume valor 255 quando conversão não tem êxito.
V 4.2	09/11/1995	Saídas da Expansão devem permanecer desligadas no caso de falta de energia.
V 4.4	14/05/1996	Corrigido erro de reset por watch-dog via DXNET. Aumentados delay para interface de Expansão.
V 5.3	25/06/1997	Corrigida rotina de delay para gravação de EEPROM.
V 5.7	04/01/1998	Retirado forçamento de variável do PWM para 255 (necessário para conversor A/D).

V 6.7	02/08/1999	Propagação de carry. Instrução adicional para acesso direto aos bits de variáveis. Corrigido erro ao escrever em variável quando falta energia elétrica.
V 6.8	15/08/1999	Versão Turbo (16.777MHz). Corrigido erro de perda de variáveis ao desligar µDX com pilhas. Proteção adicional ao decodificar instrução PARAR.
V 6.9	23/10/2000	Inclui opções SET e RST para acionar saídas da Expansão diretamente.
V 7.0	14/05/2001	Corrigido atraso no relógio (RTC) ao usar blocos DXNET.
V 7.1	05/10/2001	Corrigido bug no nodo 62 (GND).
V 7.2	07/11/2001	Corrigido bug no nodo ED (energia desliga).

Biblioteca de blocos para µDX100

A biblioteca de blocos contém todos os blocos de programação para o controlador µDX100, acessíveis via Editor PG. Abaixo as versões existentes até a data de confecção deste manual.

V 1.0	21/09/2010	Versão inicial com 23 blocos.
V 1.1	13/08/2012	Suporte à simulação, com representação de blocos energizados.
V 1.2	07/10/2015	Operações aritméticas e de comparação em Word (25 blocos). Somente operacionais para µDX101 versão 9.6 ou superior.

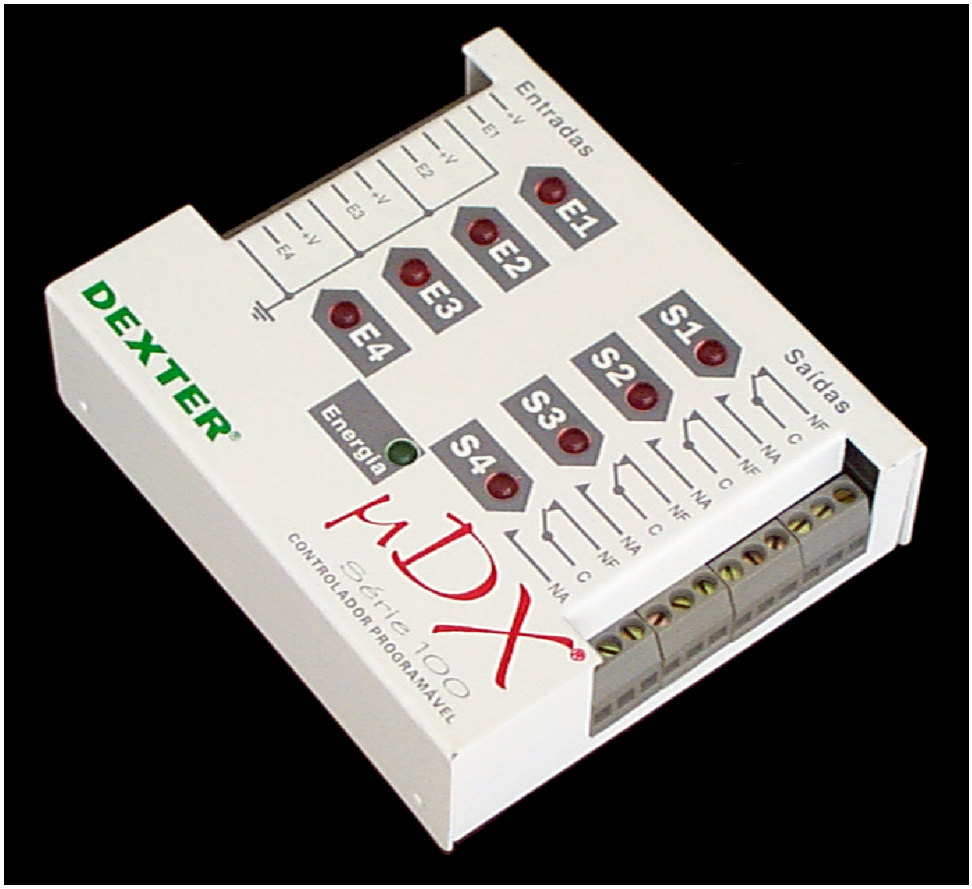
Parte II

Controlador µDX100 Plus

O controlador µDX100 Plus (µDX100+) é idêntico, externamente, ao controlador µDX100, sendo que trata-se apenas de uma versão com mais memória de programação e variáveis, além de permitir maior número de Expansões de Entradas/Saídas.

As melhorias do µDX100 Plus em relação ao µDX100 são as seguintes:

- Dobro de blocos de instruções (256 blocos).
- Quádruplo de variáveis (64 variáveis de 8 bits).
- Triplo de nodos (192 nodos).
- Conexão de até 4 Expansões de Entradas/Saídas (perfazendo 36 entradas e 36 saídas).
- Entrada de contagem rápida via entrada E2 (até 3000Hz).



ATENÇÃO: O controlador µDX100 Plus não é recomendado para novas aplicações. Neste caso é aconselhável o uso de µDX101, que possui muito mais recursos e permite migrar diretamente os programas aplicativos..

Especificações Técnicas

Características Gerais

- 52 Instruções, incluindo lógica e aritmética de 8 bits.
- 64 timers de 8 bits (compartilhados com as variáveis).
- 64 variáveis de 8 bits.
- Estruturação do programa por rede nodal.
- Execução do programa em modo de paralelismo lógico.
- Memória com capacidade para 256 blocos de instruções e 192 nodos.
- Quatro durações de ciclo: 1/16, 1/32, 1/64 e 1/256 segundo.
- "Watch-Dog-Timer".
- 4 entradas e 4 saídas digitais, expansível para 36 entradas e 36 saídas.
- Entrada de contagem rápida até 3000 Hz.
- 3 entradas analógicas (por PWM).

Características Elétricas

- Alimentação (ENERGIA): 9-14VDC @ 250mA (máximo)
- Bateria Interna: 4 pilhas tipo AA (ou baterias recarregáveis)
- Consumo na alimentação pela bateria: aproximadamente 5mA.
- Oscilador Central: 16,777216 MHz.
- Temperatura de operação: 0°C até 60°C.
- Entradas:
    - Lógica normalmente a zero (com "pull-down" de 10K).
    - Detecção de 1 lógico: +2.0 até +48.0V
    - Detecção de 0 lógico: -48.0V até +0.9V.
    - Tensão máxima na entrada (1 minuto): 120VCA.
    - Freqüência de entrada: DC - 10Hz (260KHz para bloco PWMin).
    - Tensão em +V (em aberto): 9-14VDC (conforme ENERGIA)
- Saídas:
    - Tipo de saída: 1 reversor (por cada saída)
    - Tensão nos contatos: 30VDC/220VCA
    - Corrente máxima: 10A
    - Vida útil sem carga: 100.000.000 operações
    - Vida útil com carga resistiva máxima (CA): 80.000 operações
    - Isolação (entre bobinas e contatos): 2.000Vef (1 minuto)
- Conector de Expansão:
    - Saída de alimentação +V: máx. 100mA (9 a 14 VDC)
    - Saída de alimentação +5V: máx. 8mA (+V=9VDC); máx. 50mA (+V=14VDC)

- Saída de alimentação VCC: máx. 5mA (consumo direto das pilhas)

Versões de Software

Firmware do µDX100+

Trata-se do software interno do controlador µDX100+. Sua versão pode ser obtida ao ler o Status do controlador.

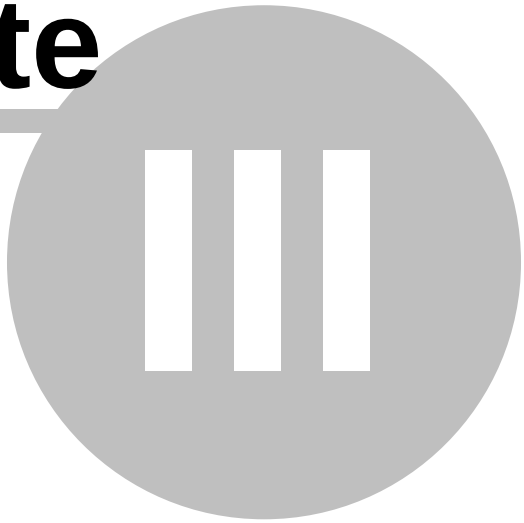
V 8.0	23/09/2001	Primeira versão operacional do µDX100 Plus. Corrigido bug no nodo 62 (GND).
V 8.1	03/04/2002	DXNET expandida para 256 endereços (16 conjuntos de 16 endereços DXNET). Entrada E2 com contagem rápida. Status (comando DXNET 0Bh) retorna conjunto DXNET.
V 8.2	24/07/2002	Permite até 4 Expansões de I/O (36 entradas & 36 saídas).
V 8.3	19/04/2003	Correção nos temporizadores (não devem atrasar devido a comunicação DXNET).
V 8.4	12/08/2003	Correção nos blocos EL & ED (Energia Liga & Energia Desliga).
V 8.5	04/04/2004	Blocos DXNET consideram endereçamento estendido (VAR>15).
V 8.6	29/10/2006	Blocos DXNET consideram conjunto no endereço DXNET de destino.
V 8.7	12/08/2008	Faltou inicializar blocos DXNET com conjunto zero.
V 8.8	15/05/2018	Novo processador baseado em FLASH e não EEPROM OTP.

Biblioteca de blocos para µDX100+

A biblioteca de blocos contém todos os blocos de programação para o controlador µDX100+, acessíveis via Editor PG. Abaixo as versões existentes até a data de confecção deste manual.

V 1.0	21/09/2010	Versão inicial com 23 blocos.
V 1.1	13/08/2012	Suporte à simulação, com representação de blocos energizados.
V 1.2	07/10/2015	Operações aritméticas e de comparação em Word (25 blocos). Somente operacionais para µDX101 versão 9.6 ou superior.

Parte



Controlador µDX101

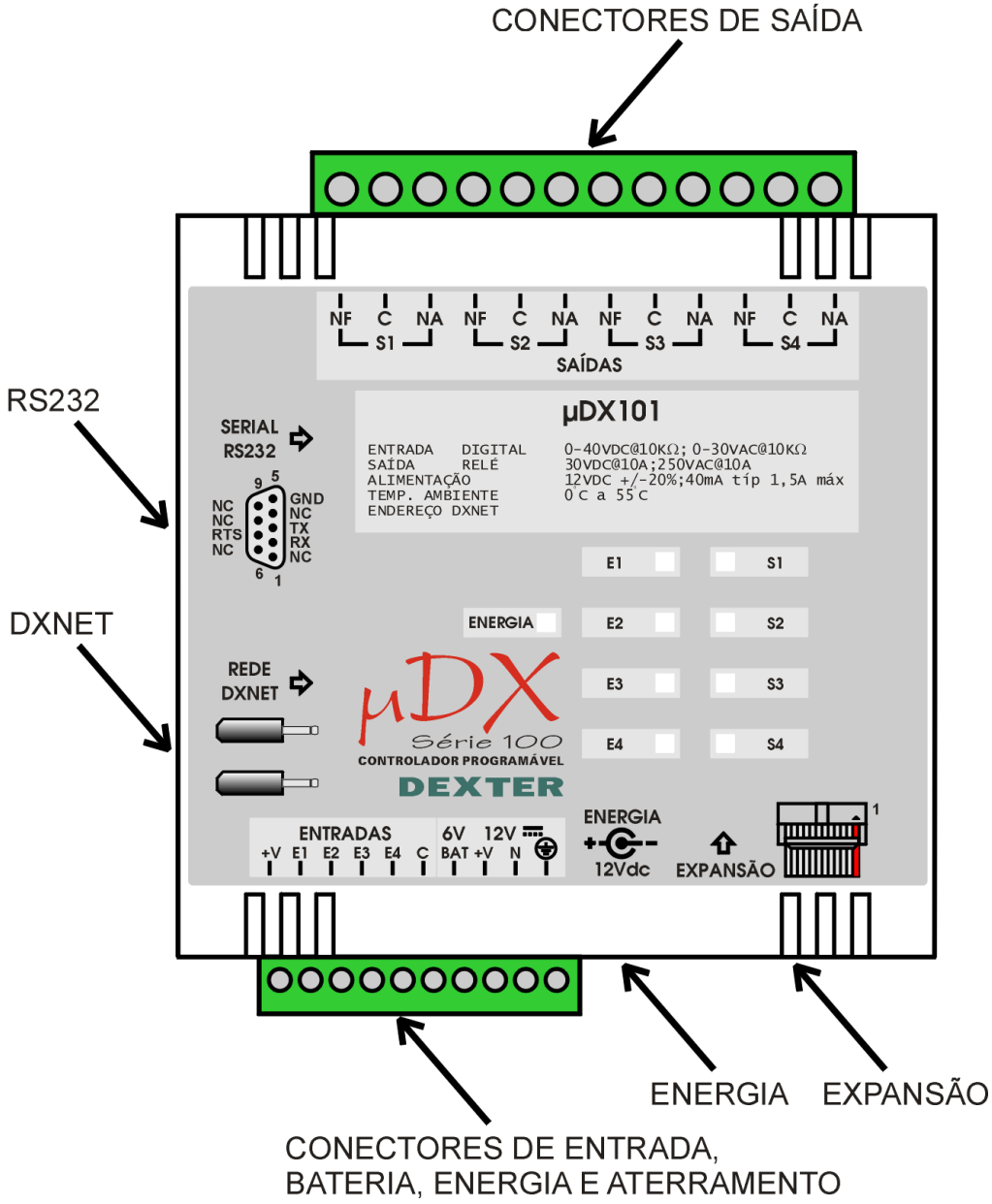
O Controlador µDX101 é uma nova versão de controlador programável da série µDX100, com acréscimo de porta serial RS232. Isso permite sua interface com o software PG em Windows sem a necessidade de unidade de Modem para µDX100. Ele mantém todas as características já consagradas da série µDX100, como 4 entradas digitais e 4 saídas digitais com relés de potência, em um gabinete mais compacto, além de vários recursos adicionais, como leitura de entrada analógica e operações em 16 bits.

As melhorias do µDX101 em relação ao µDX100 Plus são as seguintes:

- Porta serial RS232 para programação do CLP.
- Quádruplo de variáveis (256 variáveis de 8 bits).
- Limite de blocos de flip-flop (FF) expandido para 64 blocos.
- Limite de blocos DXNET expandido para 64 blocos.
- Conectores de engate rápido para entradas e saídas.
- Alimentação de bateria por bateria externa.
- Leitura de tensão analógica aplicada a entrada E1 (4,096V com 40mV de resolução - 10 bits).
- Blocos aritméticos e de comparação em word (16 bits sem sinal).
- Operações de multiplicação e divisão em word (16 bits sem sinal).
- Montagem flexível, tanto pelo fundo da caixa quanto pela lateral, permitindo economia de espaço no painel.

Conectores

Os conectores do µDX101 são os seguintes:



Entradas

São quatro entradas (E1 até E4), disponíveis em conector de engate rápido. Existe um terminal de +V (12V) e um terminal C (comum). É possível alimentar algum circuito externo utilizando-se o +V e o C (comum) para isso. O limite de corrente disponível no +V é 100mA.

> **Atenção:** não existe isolação galvânica entre qualquer um dos terminais de entrada e o restante do circuito interno do µDX101. Assim, estas entradas não podem ser conectadas diretamente à rede elétrica. Para isso deve-se usar o módulo de Opto-acoplador.

Saídas

São quatro saídas de três terminais cada um, disponíveis em conector de engate rápido. Como indicado no policarbonato do painel do µDX101, o segundo terminal de cada saída corresponde ao contato móvel interno do relé de saída. Os outros dois terminais referem-se aos contatos dos relés, Normal Fechado e Normal Aberto.

Os relés possuem isolação galvânica entre si e entre os contatos e o circuito interno do µDX101. Não existe interconexão interna entre os relés. Portanto, eles podem ser usados para comandar diretamente cargas ligadas à rede elétrica.

Energia

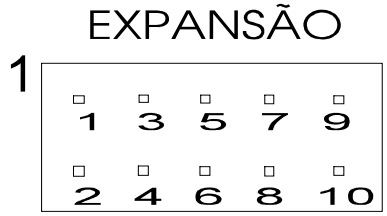
Para alimentação do µDX101 pode ser usado o conector P4 existente na lateral do controlador, ou os pinos +V e N (neutro) presentes no conector de entrada. Acompanha o controlador uma pequena fonte de alimentação já própria para conexão ao conector P4 do µDX101. A energia externa para alimentar o µDX101 deve ser entre 9 e 14 VDC suprindo até 250mA. Como representado no desenho do policarbonato, o contato central é ligado ao comum (Zero Volts). O contato externo é ligado aos 12V, que alimenta os circuitos dos relés, a fonte interna que reduz a tensão para 5 Volts e ao terminal +V do conector de entrada.

Bateria

Está disponível no conector de entrada o pino BAT, que permite a conexão de bateria externa ao µDX101, de forma a manter o relógio de tempo real e o funcionamento do CLP no caso de falta de energia elétrica. Os relés de saída ficam inoperantes enquanto a alimentação estiver sendo suprida por este pino, mas as demais funções do controlador, como rede DXNET, porta serial RS232 e execução do programa aplicativo continuam operando. A bateria deve ser conectada entre o pino BAT (terminal positivo) e o terninal N (neutro) do conector de entrada. Esta bateria deve suprir entre 4,5 e 6V em seus terminais. O usual é usar 4 pilhas AA. Se forem do tipo recarregável é possível fechar um jumper interno no µDX101, de forma a permitir uma pequena corrente de carga para as pilhas recarregáveis quando o controlador estiver energizado. O consumo sob pilhas é de cerca de 5mA.

Expansão

É um tipo barra de pinos, com 10 contatos em duas filas de 5. O primeiro fica no canto superior esquerdo, sendo a numeração dos contatos a seguinte:



Serve para conectar ao µDX101 algum circuito externo para a entrada e saída de dados de 8 bits, como a Expansão de Entradas/Saídas.

Significado dos contatos:

1 - Comum (0V)
2 - +V (diretamente ligado à energia)
3 - Pulso de carga ("Load")
4 - +5V (máx. 10mA)
5 - Pulso de relógio ("Clock")
6 - Comum (0V)
7 - Entrada de dado serial ("Din")

8 - VCC (+5V garantidos pelas pilhas quando falta energia, máx. 5 mA)
9 - Saída de dado serial ("Dout")
10 - Comum (0V)

DXNET

São dois conectores tipo P2, fêmea, que estão internamente ligado em paralelo. Em qualquer um dos dois podem ser conectados os cabos para rede DXNET ou o cabo do adaptador para a comunicação com o computador. O contato da ponta é o sinal e o contato do corpo é o comum (Zero Volts, "Ground"). O sinal está ligado diretamente ao microcontrolador do µDX101 e a um resistor de 1,8K (que vai ligado ao VCC).

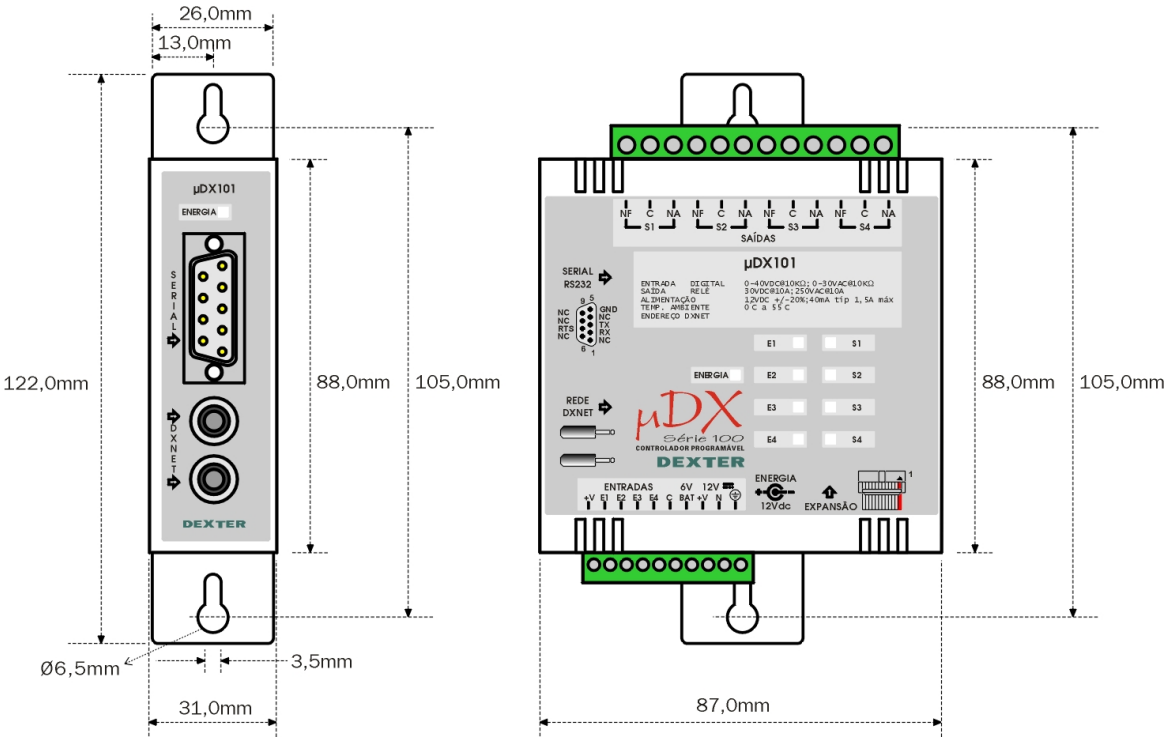
> Cuidado para não aplicar no sinal tensões superiores à +5 Volts, nem inferiores à -0,5 Volts, sob risco de dano irreversível do microcontrolador.

RS232

O conector RS232 permite programar o controlador µDX101 através de um computador IBM-PC compatível, rodando Windows (de Windows 98 até Windows 7). Caso seu computador não possua porta serial RS232 (situação muito comum em notebooks) pode ser usado um cabo adaptador USB-RS232, de forma a programar o µDX101 via porta USB. A Dexter comercializa este tipo de cabo. Acompanha o controlador µDX101 um cabo serial para conexão ao computador.

Fixação Mecânica

Acompanha o controlador µDX101 um suporte para parafuso, que permite fixá-lo tanto pelo fundo da caixa quanto pela lateral da mesma. Esta segunda forma de fixação permite grande economia de área, como ilustrado a seguir:



Especificações Técnicas

Características Gerais

- 68 Instruções, incluindo lógica e aritmética de 8 bits e 16 bits.
- 256 timers de 8 bits (compartilhados com as variáveis).
- 256 variáveis de 8 bits.
- Estruturação do programa por rede nodal.
- Execução do programa em modo de paralelismo lógico.
- Memória com capacidade para 256 blocos de instruções e 192 nodos.
- Quatro durações de ciclo: 1/16, 1/32, 1/64 e 1/256 segundo.
- "Watch-Dog-Timer".
- 4 entradas e 4 saídas digitais, expansível para 36 entradas e 36 saídas.
- Entrada de contagem rápida até 3000 Hz.
- Entrada analógica com fundo de escala em 4,096V e resolução de 40mV.
- 3 entradas analógicas (por PWM).

Características Elétricas

- Alimentação (ENERGIA): 9-14VDC @ 250mA (máximo)
- Bateria externa: 4 pilhas tipo AA (ou baterias recarregáveis)
- Consumo na alimentação pela bateria: aproximadamente 5mA.
- Oscilador Central: 16,777216 MHz.
- Temperatura de operação: 0°C até 60°C.
- Entradas:
    - Lógica normalmente a zero (com "pull-down" de 10K).
    - Detecção de 1 lógico: +2.0 até +30.0V
    - Detecção de 0 lógico: -30.0V até +0.9V.
    - Tensão máxima na entrada (1 minuto): 120VCA.
    - Freqüência de entrada: DC - 10Hz (260KHz para bloco PWMin).
    - Tensão em +V (em aberto): 9-14VDC (conforme ENERGIA)
- Saídas:
    - Tipo de saída: 1 reversor (por cada saída)
    - Tensão nos contatos: 30VDC/220VCA
    - Corrente máxima: 10A
    - Vida útil sem carga: 100.000.000 operações
    - Vida útil com carga resistiva máxima (CA): 80.000 operações
    - Isolação (entre bobinas e contatos): 2.000Vef (1 minuto)
- Conector de Expansão:
    - Saída de alimentação +V: máx. 100mA (9 a 14 VDC)

- Saída de alimentação +5V: máx. 8mA (+V=9VDC); máx. 50mA (+V=14VDC)
- Saída de alimentação VCC: máx. 5mA (consumo direto das pilhas)

Versões de Software

Firmware do µDX101

Trata-se do software interno do controlador µDX100+. Sua versão pode ser obtida ao ler o Status do controlador.

V 9.0	27/04/2012	Primeira versão operacional do µDX101.
V 9.1	07/08/2013	Previsto forçamento de baud-rate em 38400 se memória de programa limpa (com FFh).
V 9.2	11/08/2013	Evita desligamento dos relés do µDX101 quando entra em stop.
V 9.3	19/08/2014	Retirada modificação introduzida na versão 9.2, pois impedia que os relés fossem acionados com o programa aplicativo parado. Nova estratégia para evitar desligamento dos relés ao parar programa aplicativo.
V 9.4	25/08/2014	Entrada analógica (10 bits) em E1 (lida em conjunto com entrada de contagem rápida em E2).
V 9.5	15/12/2014	Utilizada nova ferramenta de compilação.
V 9.6	07/10/2015	Operações aritméticas e de comparação em Word (16 bits sem sinal). Operação de multiplicação e divisão em Word (16 bits sem sinal). Adequação de baud-rate conforme clock interno usado (16 ou 16,777MHz). Verificação de DXNET de forma a evitar trancamento do CLP por DXNET em nível zero.
V 9.7	23/03/2016	Corrigido erro que alterava nodos acima de n63 no caso de monitoramento de variáveis altas (acima de v63).
V 9.8	04/04/2016	Melhorias na detecção de falha de alimentação elétrica.
V 9.9	07/02/2017	Correção na leitura/escrita de variáveis word (16 bits) via DXNET.

Biblioteca de blocos para µDX101

A biblioteca de blocos contém todos os blocos de programação para o controlador µDX101, acessíveis via Editor PG. Abaixo as versões existentes até a data de confecção deste manual.

V 1.0	21/09/2010	Versão inicial com 23 blocos.
V 1.1	13/08/2012	Suporte à simulação, com representação de blocos energizados.
V 1.2	07/10/2015	Operações aritméticas e de comparação em Word (25 blocos). Somente operacionais para µDX101 versão 9.6 ou superior.

Editor PG para µDX101

O programa Editor PG permite gerar os arquivos fonte para os programas aplicativos do controlador µDX101.

1.0.0.5	Set/2010	Primeira versão de publicação.
1.1.0.0	Mai/2011	Correção na alocação de variáveis auxiliares em Macros. Nodos ED e EL são contados como blocos no Editor. Correção na faixa de tempo de Oscilador em minutos. Correção na indicação de µDX100 Turbo ou Plus no Compilador. Cálculo do endereço de constantes na IHM corrigido. Erro no tamanho da representação de Macro corrigido. Previsão para controlador programável µDX101. Janelas de periféricos mantêm dados ao reabri-las.

1.1.0.1	Dez/2011	Leitura incorreta do conjunto 14 de nodos (N114 a N119) na monitoração. Diversos blocos de saída e diversos blocos de entrada não compilam corretamente para o mesmo nodo. Lançamento incorreto dos bits complementares na compilação para uDX 101. Inclui velocidades 38400 e 19200 nas opções da janela de busca pelo uDX (RS232). Inclui opção para busca apenas pelo endereço zero (ou todos) na janela de busca pela serial. Corrigido o limite máximo de blocos DXNET (8).
1.1.0.2	Abr/2012	Otimização no tempo de timeout caso o µDX100 não responda. Adaptação para verificação de versão no novo servidor do web site Dexter.
2.0.0.0	Ago/2012	Simulador para µDX100, µDX100 Plus e µDX101.
2.0.0.1	Dez/2012	Presença do nodo EL, em certos casos, gera falha de compilação.
2.0.0.2	Jun/2013	Correções no simulador.
2.0.0.3	Ago/2013	Transmissão de dados da Configuração de Hardware mesmo tratando-se de µDX100/µDX100+. Resolução do Sensor de Umidade no Conversor A/D deve ser 1 e não 100.
2.0.0.4	Ago/2014	Corrigido erro na simulação de bloco Relógio.
2.0.1.0	Out/2015	Implementação da compilação, monitoração e simulação de funções word. Device padrão para cofigurações iniciais (na instalação) passa a ser µDX101. Funções word operacionais apenas para µDX101 versão 9.6 ou superior.
2.0.2.0	Mar/2016	Representação de controlador µDX101 e expansões µDX110 na tela do Compilador. Correções na simulação de blocos de função e comparação Word. Representação de entradas/saídas para Expansão µDX101 de 1 a 4, em vez de 5 a 12 como no caso da Expansão do µDX100.
2.0.2.2	Abr/2017	Prevista leitura de variáveis 16 bits (word) via IHM.
2.0.2.3	Abr/2018	Velocidade de transmissão de programa aplicativo para µDX101 melhorada significativamente.
2.1.0.0	Ago/2020	Compilação de IHM prevê edição de constantes em 16 bits em vez de de seleção entre µDX100 Turbo ou µDX100 Plus / µDX101. Note que essa funcionalidade exige IHM para µDX100 v2.7 ou superior.
2.1.0.1	Mai/2021	Retirada restrição de constante < 128 para edição em 16 bits na IHM.
2.1.1.0	Jun/2021	Correção na simulação de reset de Flip-Flop.

Compilador PG para µDX101

O programa Compilador PG permite compilar os arquivos fonte gerados pelo Editor PG, assim como efetuar comunicação serial com o controlador µDX101.

1.01	Set/2010	Versão inicial.
1.02	Mai/2011	Diversas correções na compilação dos blocos do µDX100.
1.03	Mai/2011	Suporte a controlador µDX101.
1.04	Ago/2013	Transmissão de dados da Configuração de Hardware mesmo tratando-se de µDX100/µDX100+. Resolução do Sensor de Umidade no Conversor A/D deve ser 1 e não 100.
1.05	Set/2014	Corrigido bug gerado por bloco de Energia sem conexão.
1.06	Jul/2015	Previsão de blocos de Função/Comparação Word (16 bits).

1.07	Out/2015	Implementação de blocos de Função/Comparação Word (16 bits), incluindo alocação 16 bits (word) para variáveis. Funções word operacionais apenas para µDX101 versão 9.6 ou superior.
1.08	Mar/2016	Correções na compilação de blocos Word. Representação de entradas/saídas para Expansão µDX101 de 1 a 4, em vez de 5 a 12 como no caso da Expansão do µDX100.
1.09	Ago/2020	IHM com previsão de edição de constantes em 16 bits. Note que essa funcionalidade exige IHM para µDX100 v2.7 ou superior.

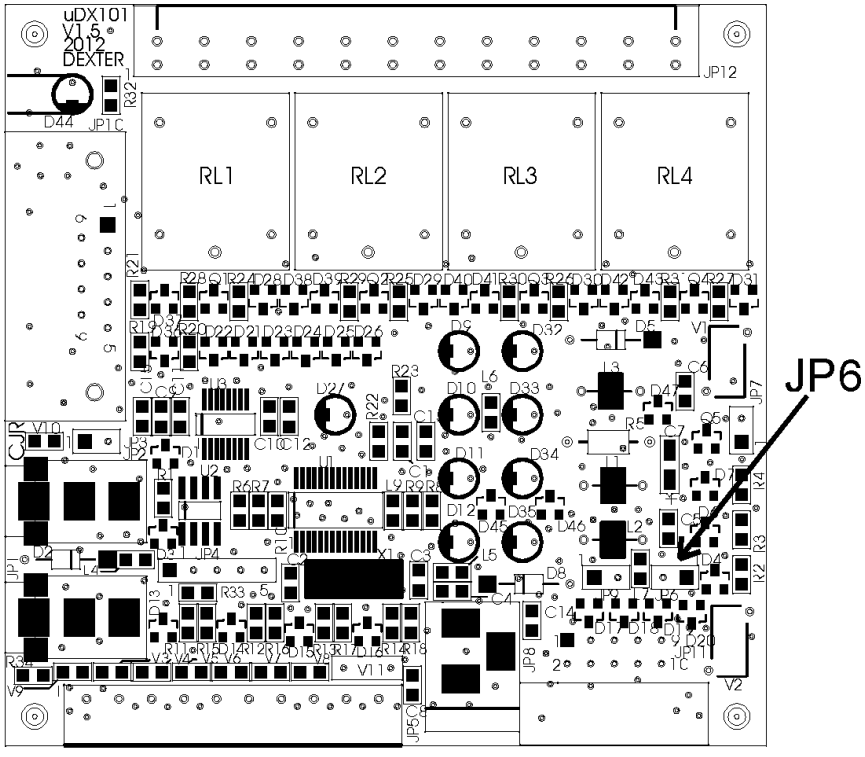
Jumpers

O único estrape (jumper) existente internamente ao controlador µDX101 serve para comutar entre pilhas comuns ou pilhas recarregáveis. No caso do jumper JP6 estar fechado o próprio µDX101 fornece uma pequena corrente de recarga ao conjunto de pilhas, mantendo-as carregadas, enquanto estiver sendo alimentado pela rede elétrica.

Note que só é necessário usar pilhas (conjunto de quatro, perfazendo entre 4,8 e 6V) caso o programa aplicativo que roda no µDX101 faça uso do relógio de tempo real do controlador (por exemplo, para efetuar tarefas em determinados horários). Neste caso as pilhas irão manter o relógio operando no caso de interrupção no fornecimento de energia elétrica, evitando que o mesmo perca o horário programado.

Não se deve manter o µDX101 operando sob pilhas durante um tempo excessivo, nem armazená-lo com pilhas alimentando-o. Isso porque o CLP consome uma corrente razoável das pilhas quando não está alimentado pela rede elétrica, e pode esgotá-las em cerca de 15 a 20 dias.

A posição do jumper JP6 é mostrado no desenho a seguir:



Placa impressa do µDX101

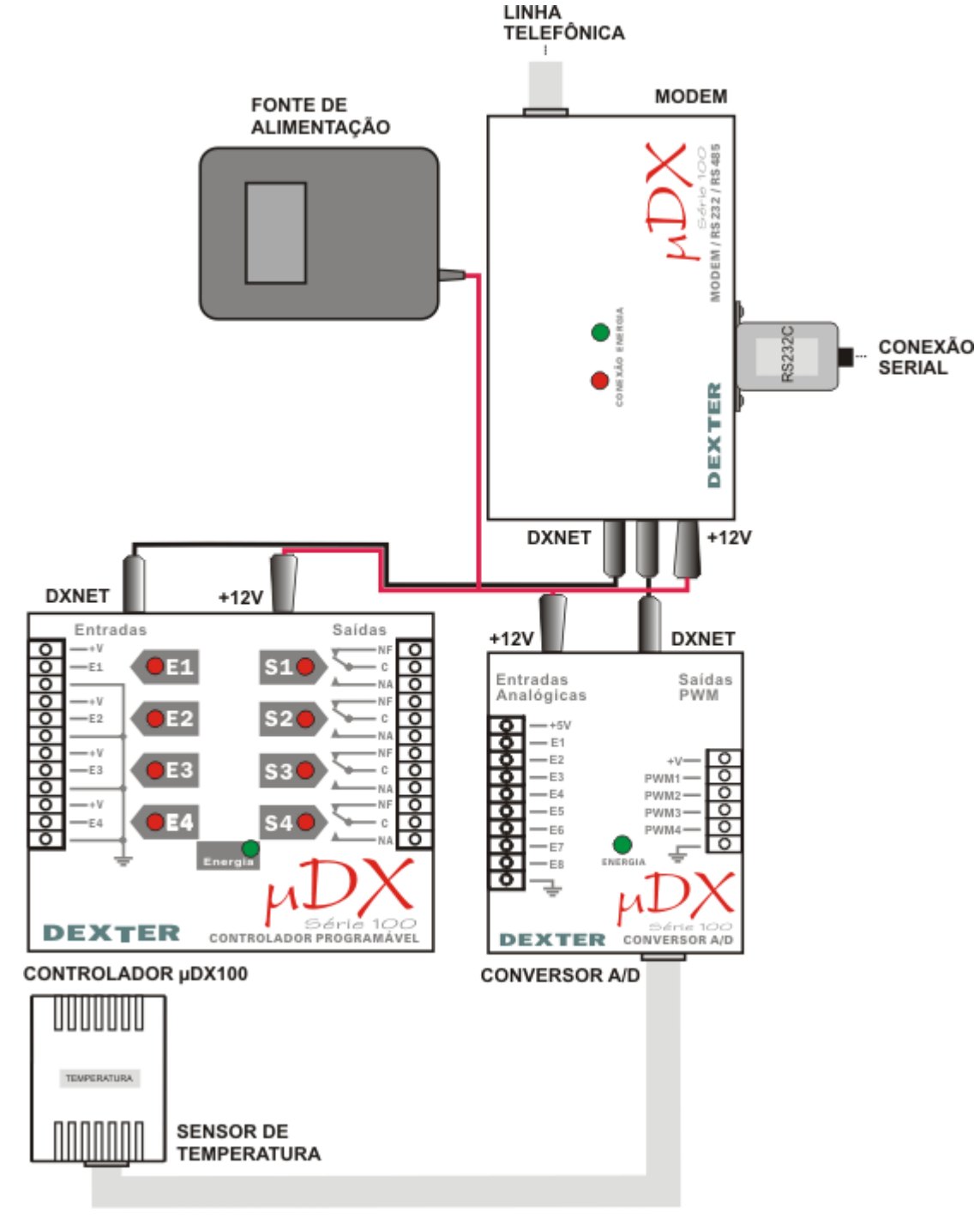
Para abrir a caixa metálica do µDX101 retire os dois parafusos (fenda cruzada) existentes nas

laterais da caixa, e force levemente as laterais para que se afastem dos encaixes que a prendem ao fundo da caixa.

Parte

Periféricos

O Controlador µDX100 permite a conexão direta de Expansão de Entradas/Saídas (no caso de µDX100+ até 4 Expansões), e diversos periféricos via rede DXNET, como Interface Homem/Máquina (IHM), Conversor A/D e Modem.



Expansão de Entradas/Saídas

Este documento descreve as características da expansão de entradas/saídas do µDX (8 entradas e 8 saídas). Com esta expansão, o número de pontos de entradas e saídas do µDX sobe para 12 de cada (8 entradas expansão + 4 entradas µDX e 8 saídas expansão + 4 saídas µDX). Isto aumenta muito a gama de possíveis aplicações do equipamento.

Além disso, as entradas da expansão são opto-acopladas, permitindo ligação direta a rede elétrica (127 ou 220 VAC) e oferecendo alta imunidade contra ruídos. Já as saídas da expansão utilizam relés (iguais aos utilizados no µDX), com capacidade até 10 A (carga resistiva).

A expansão é conectada ao µDX através de um pequeno cabo chato (flat-cable) de 10 vias, fornecido juntamente com o equipamento. O suprimento de energia elétrica para a expansão é retirado da fonte de alimentação do µDX a ela conectado. A fonte do µDX está dimensionada para este consumo adicional.

No caso do controlador µDX+ é possível conectar até 4 Expansões de Entradas/Saídas, perfazendo 32 entradas e 32 saídas adicionais (com as entradas e saídas do próprio controlador o total chega a 36 entradas e 36 saídas).

Conexão ao µDX

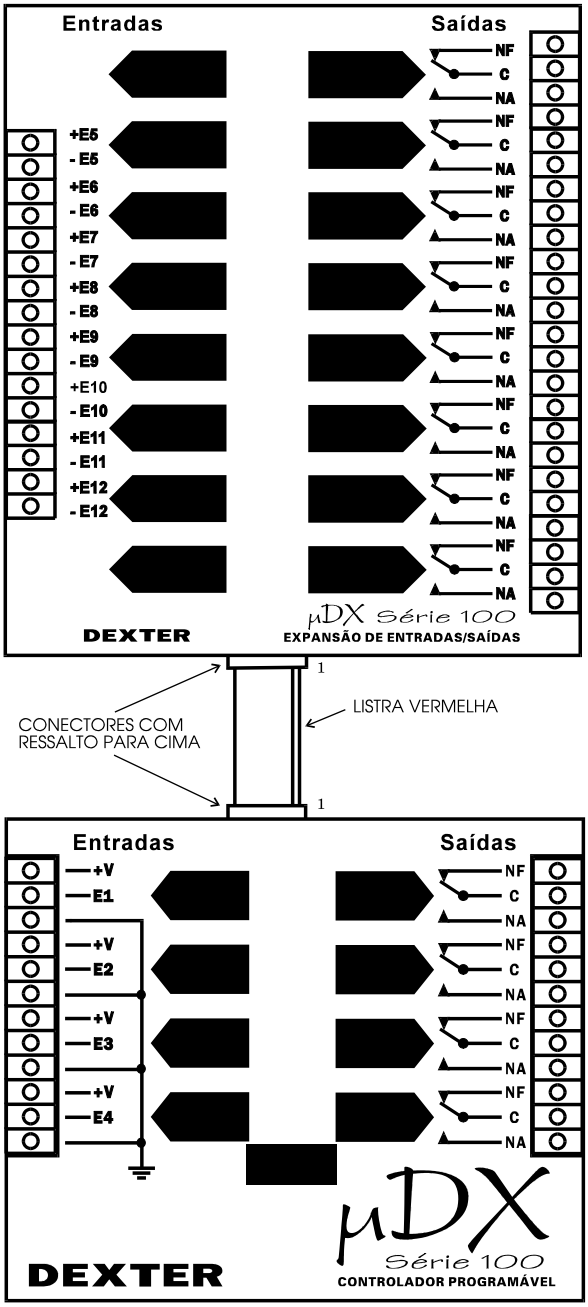
Acompanha a expansão um pequeno cabo chato ("flat-cable") de 10 vias para sua conexão ao µDX. Este cabo já é remetido conectado à expansão. A extremidade livre deve ser conectada ao conector "EXPANSÃO" existente na lateral da caixa do µDX. Note que o pino 1 deste conector liga com pino 1 do conector existente na lateral da expansão de entradas/saídas. O cabo possui uma tarja vermelha para designar o lado do pino 1. Além disso, os conectores em suas extremidades possuem um ressalto, permitindo a inserção apenas com este ressalto para cima.

Atenção: Não force a entrada do conector na expansão ou no µDX. Caso ofereça resistência à conexão, verifique se o ressalto não está voltado para o fundo do equipamento. Neste caso vire o cabo, conectando-o com os ressaltos para cima.

A alimentação elétrica da expansão de entradas/saídas é suprida pelo µDX, através do cabo de conexão. Sempre que o µDX100 sofrer um reinicialização ("reset") as saídas da expansão serão desativadas.

Uma vez conectada, a expansão já está apta a operar. Ela possui 16 indicadores luminosos ("leds") que indicam a energização de suas entradas e saídas.

Abaixo temos um desenho da expansão conectada ao µDX100. O cabo de conexão possui cerca de 7 cm, o que limita a distância máxima entre µDX e expansão. Na prática, recomenda-se uma distância de 4,5 cm, de forma a poder conectar o cabo de DXNET e o cabo de alimentação do µDX100 com tranqüilidade.



Atenção: No caso do controlador µDX+ é possível conectar até 4 Expansões de Entradas/Saídas, perfazendo 32 entradas e 32 saídas adicionais (com as entradas e saídas do próprio controlador o total chega a 36 entradas e 36 saídas). Para isso é necessária a utilização do novo modelo de Expansão, com dois conectores de expansão. Basta conectar uma expansão na outra. A Expansão mais próxima do controlador irá assumir as entradas E5 a E12 e saídas S5 a S12; a seguinte será as entradas 2E-E5 a 2E-E12 e saídas 2S-S5 a 2S-S12; e assim por diante.

ATENÇÃO: A fonte de alimentação que acompanha o controlador µDX+ possui capacidade de corrente para suprir até 12 relés acionados simultaneamente (4 do µDX+ e 8 de uma Expansão). No caso de uso de múltiplas Expansões é necessário verificar se o programa aplicativo não excede esta capacidade em algum momento ou substituir a fonte de alimentação por uma mais potente. Para acionar todos os 36 relés do µDX+ e das 4 Expansões a fonte deve suprir 12V @ 1,5A (18 W).

Entradas

As entradas da expansão de entradas/saídas do µDX100 são todas opto-isoladas, oferecendo isolação galvânica. Assim, é possível conectá-las diretamente à rede elétrica (127 ou 220 Vac) ou usar várias fontes de sinal, sem conexão de referência (terra) entre elas.

A designação dada as entradas é E5 a E12.

Internamente a expansão de entradas/saídas existem diversos "jumpers" removíveis, que permitem configurar individualmente as oito entradas. Para acessá-los é necessário abrir a caixa do equipamento.

Desligue o cabo de conexão da expansão de entradas/saídas (com este cabo conectado não é possível abrir a caixa). Abra a caixa da expansão forçando levemente as laterais para afastarem-se dos encaixes que prendem a tampa ao fundo. Puxe a tampa cuidadosamente para cima. Cuidado com os LEDs que estão montados presos a pequenos conectores de dois contatos. Caso algum dos LEDs saia do lugar observe a posição dos LEDs adjacentes para saber como encaixar o que saiu.

Os "jumpers" internos permitem configurar as entradas da expansão para quatro tipos de sinal:

1) Alta Tensão AC	**110 a 220 VAC (corrente alternada de 50 ou 60 Hz)**
2) Alta Tensão DC	**110 a 220 VDC (corrente contínua)**
3) Baixa Tensão AC	**6 a 30 VAC (corrente alternada de 50 ou 60 Hz)**
4) Baixa Tensão DC	**6 a 30 VDC (corrente contínua)**

Atenção: Caso alguma entrada esteja preparada para a opção de baixa tensão (6 a 30 V), a conexão direta a rede elétrica (127 ou 220 VAC) provocará sua queima instantânea.

Normalmente a expansão é remetida com os "jumpers" instalados para a opção 1 (alta tensão AC), pois neste caso evita-se a queima acidental de uma das entradas ao ligá-la direto à rede elétrica.

Já a situação contrária, ou seja, ligar em 6 a 30 V uma entrada programada para 110 a 220 V, não traz maiores conseqüências (embora a entrada não consiga ser energizada por uma tensão tão baixa).

Na placa da expansão existem duas colunas de "jumpers", numerados de JP14 a JP21 e de JP22 a JP29. Estes jumpers permitem comutar o modo de operação das entradas da expansão. A coluna de jumpers JP14 a JP21 comuta entre alta tensão (jumper aberto) e baixa tensão (jumper fechado). Já a coluna JP22 a JP29 comuta entre corrente contínua (jumper aberto) e corrente alternada (jumper fechado). Assim, estando o jumper na coluna JP14 a JP21 fechado a entrada correspondente estará programada para baixa tensão. Com este jumper aberto a entrada será para alta tensão. Já se o jumper correspondente a mesma entrada estiver instalado na coluna JP22 a JP29 a entrada estará preparada para corrente alternada.

Note que uma entrada para corrente alternada (baixa ou alta tensão) pode ser acionada por uma corrente contínua sem nenhum problema (apenas ocorrerá um leve atraso para acionar ou desacionar a entrada (cerca de 50 ms) devido a filtragem). A recíproca não é verdadeira, ou seja, uma entrada programada para corrente contínua, ao ser excitada com corrente alternada, passará a ligar e desligar aleatoriamente (depende do ciclo de amostragem do µDX100 coincidir com o ciclo da rede elétrica).

A seguir temos uma tabela com a programação para todas as entradas da expansão (1 indica jumper fechado e 0 indica jumper aberto):

		Alta Tensão		Baixa Tensão	
Entrada	**Jumpers**	**DC**	**AC**	**DC**	**AC**
E5	JP21,JP29	00	01	10	11
E6	JP20,JP28	00	01	10	11
E7	JP19,JP27	00	01	10	11
E8	JP18,JP26	00	01	10	11
E9	JP17,JP25	00	01	10	11
E10	JP16,JP24	00	01	10	11
E11	JP15,JP23	00	01	10	11
E12	JP14,JP22	00	01	10	11

Por exemplo, se quisermos a entrada E8 em alta tensão AC devemos abrir o jumper JP18 e fechar o jumper JP26. Para programar E11 como baixa tensão AC devemos fechar JP15 e JP23 e assim por diante.

Note que no caso de alta tensão pode-se ligar a entrada tanto em rede de 110 VCA quanto 220 VCA. No caso de rede de 110 VCA (ou 127 VCA, como é comum no Brasil), o LED indicativo correspondente a entrada irá ligar com brilho menor do que quando ligado em 220 VCA. Isto é perfeitamente normal e não causa nenhum transtorno. O mesmo ocorre com entradas em baixa tensão, que ligam o LED com mais brilho em 30 V do que no limite inferior (6V).

Atenção: No caso de entrada em corrente contínua deve ser respeitada a polaridade inscrita na tampa superior do equipamento. Já em corrente alternada (50 ou 60 Hz) a polaridade perde significado.

Saídas

As saídas da expansão são constituídas de relés com um contato reversor, isto é, possui um contato central e dois adjacentes, ficando um deles ligado e outro desligado quando a saída está inativa. Quando a saída fica ativa o contato central troca de lado, invertendo os contatos ligado e desligado.

Esta particularidade permite que se ligue ou que se desligue algum dispositivo quando a saída for acionada, bastando escolher qual dos contatos adjacentes deva ser empregado. A serigrafia da tampa da caixa (lado direito da tampa) mostra estas conexões e a representação dos contatos dos relés.

A designação dada as saídas é S5 a S12.

Note que os relés dispõem de isolação galvânica entre a bobina de acionamento e os contatos. Isto torna possível ligar dispositivos à rede elétrica domiciliar ou industrial para serem atuados diretamente pelos contatos do relé, sem risco algum para a Expansão de Entradas/Saídas ou para o µDX100 conectado a ela.

Os relés permitem uma corrente máxima de 10 A (carga resistiva), tensão nos contatos até 220 VCA e isolação entre bobina e contatos de 2000 V (1 minuto). A vida útil estimada dos contatos dos relés é de 100.000.000 de operações sem carga, ou 80.000 operações sob carga resistiva máxima (10 A).

Interface Homem/Máquina

Este documento descreve as características da Interface Homem/Máquina (IHM) do controlador programável µDX100. Com esta interface, é possível modificar parâmetros do programa, como tempos dos blocos de temporização ou o valor de constantes no µDX100. Além disso, ela permite a exibição de mensagens alfanuméricas ou o valor de variáveis em um visor de cristal líquido, com 2 linhas de 16 caracteres cada (com iluminação própria).

A Interface possui 8 entradas analógicas, de 0 a 5 V, com resolução de 8 bits (256 divisões). Estas entradas são ligadas a um conversor analógico/digital (A/D), não necessitando conversão em largura de pulso (PWM), como no caso de entradas analógicas no µDX100.

A interface IHM tem capacidade de armazenar até 15 mensagens de 16 caracteres cada (em memória não volátil - E²PROM), e 16 mensagens adicionais para edição de constantes do programa no µDX100.

Cada IHM ocupa um endereço na rede DXNET, podendo acessar até 14 controladores µDX100 ligados à ela via rede DXNET. Nada impede a utilização de mais de uma IHM na mesma rede DXNET, tampouco.

Conexão ao µDX

Acompanha a Interface Homem/Máquina (IHM) um cabo blindado, com pinos P2 nas extremidades, para conectá-lo à rede local DXNET. Na IHM existem dois conectores P2 fêmea (designados como DXNET na caixa), interconectados em paralelo. Basta conectar o cabo em qualquer um dos conectores DXNET da IHM e a outra extremidade ao controlador programável µDX. Além disso, é fornecida uma fonte de alimentação. Portanto, os itens que compõem a IHM são:

- Interface Homem/Máquina para µDX Série 100
- Cabo para conexão à rede DXNET
- Manual de Utilização (este manual)
- Fonte de Alimentação

A fonte de alimentação é que supre de energia a IHM. A extremidade livre do cabinho que sai da fonte de alimentação deve ser ligada ao conector na IHM que está indicado como ENERGIA.

Atenção: Antes de ligar a fonte de alimentação na rede elétrica verifique se ela está ajustada adequadamente para a tensão da rede no local (127 ou 220 VCA).

Ao ligar a alimentação elétrica o indicador luminoso de ENERGIA (led) acende no painel da IHM. Quanto ao display, é normal que fique em branco, pois a interface não tem mensagens gravadas na sua memória para visualização.

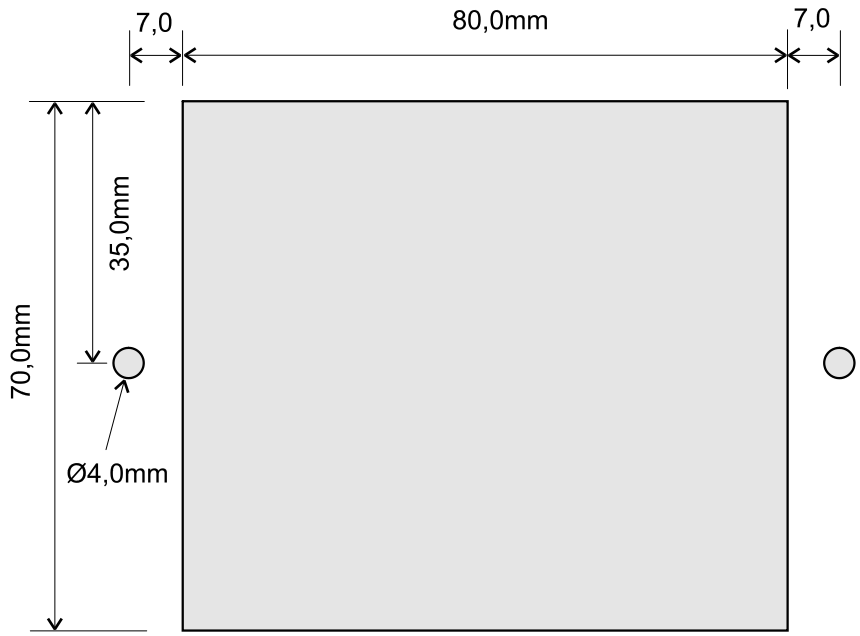
Painel Frontal

Abaixo temos uma representação do painel frontal da Interface Homem/Máquina. Este consta de um display alfanumérico de 2 linhas e 16 caracteres por linha, 4 teclas e um indicador luminoso (LED) para indicar energização.

O display normalmente apresenta as mensagens selecionadas pela variável escolhida no µDX para esta função. Ao pressionar uma das teclas de Parâmetro o display comuta para a última mensagem de edição selecionada. Usando as duas teclas de Parâmetro é possível escolher entre as 16 edições na memória da IHM. Com as teclas de Valor pode-se modificar as constantes associadas a estas mensagens de edição. Não pressionando nenhuma tecla por 4 segundos, a interface sai automaticamente de edição e volta a apresentar mensagens.

Durante a edição a linha superior do display apresenta a mensagem "Edição:", e na linha inferior a mensagem de edição. As 4 teclas da IHM têm auto-repetição.

Para fixação em painel está disponível um acessório, denominado Moldura para IHM, que permite fixar a IHM em um painel frontal e esconder o rasgo, oferecendo um acabamento esmerado. Entre em contato com a Dexter para maiores informações. A figura a seguir especifica o tamanho do rasgo a ser efetuado no painel para fixação da IHM via Moldura.



Conversor A/D

Este documento descreve as características do conversor analógico/digital para o controlador programável µDX100. O conversor A/D permite conectar 8 sinais analógicos de 0 a 10V, 0 a 5V, ou 0 a 20mA (selecionável por “jumper” interno) e 8 sensores de temperatura (-55°C a 125°C), ou ainda 4 sensores de temperatura + 4 sensores de umidade relativa (10 a 90 % UR). Os sensores de temperatura e de umidade não necessitam de nenhuma calibração ou ajuste. Para a conexão de vários sensores de temperatura ou umidade ao conversor analógico/digital a DEXTER disponibiliza uma placa de extensão. A resolução do sensor de temperatura é de 0,0156°C e sua precisão é de 0,5°C. Já as entradas analógicas tem resolução de 8 bits (256 divisões). Os sensores de umidade possuem resolução de 0,5 % UR.

O conversor A/D pode comunicar os dados lidos para até outros 14 µDXs ligados em rede de comunicação DXNET. Assim, é possível utilizar apenas um conversor A/D para todos os µDXs da rede DXNET. Nada impede a utilização de mais de um conversor A/D na mesma rede DXNET, tampouco.

Além da comunicação via rede DXNET, é possível utilizar 4 saídas em modulação de largura de pulso (PWM) para comunicação com os µDXs. Este é um processo que restringe o conversor A/D a apenas 4 variáveis analógicas (entradas analógicas ou temperaturas). Portanto, só deve ser utilizado quando não houver endereço disponível para o conversor A/D na rede DXNET.

Outra possibilidade é usar as 4 saídas PWM do Conversor A/D como saídas analógicas de 0 a 5V ou de 0 a 10V. Para isso é necessário conectar estas saídas a placas de Saídas Analógicas, comercializadas separadamente pela Dexter (Conversor A/D versão 2.1 ou superior).

O conversor é alimentado com 12Vdc, idêntico ao controlador µDX e os outros periféricos (IHM, Modem, etc.). No conector de entradas analógicas está disponível 5Vdc precisos, útil para excitar potenciômetros ou outros sensores de baixo consumo.

Conexão ao µDX

Acompanha o conversor A/D um cabo blindado, com pinos P2 nas extremidades, para conectá-lo à rede local DXNET. No conversor existem dois conectores P2 fêmea (designados como DXNET na caixa metálica do conversor A/D), interconectados em paralelo. Basta conectar o cabo em qualquer um dos conectores DXNET do conversor A/D e a outra extremidade ao controlador programável µDX. Além disso, é fornecida uma fonte de alimentação. A fonte de alimentação é que supre de energia o conversor. A extremidade livre do cabinho que sai da fonte de alimentação deve ser ligada ao conector no conversor analógico/digital que está indicado como ENERGIA.

Atenção: Antes de ligar a fonte de alimentação na rede elétrica verifique se ela está ajustada adequadamente para a tensão da rede elétrica local (127 ou 220 VCA).

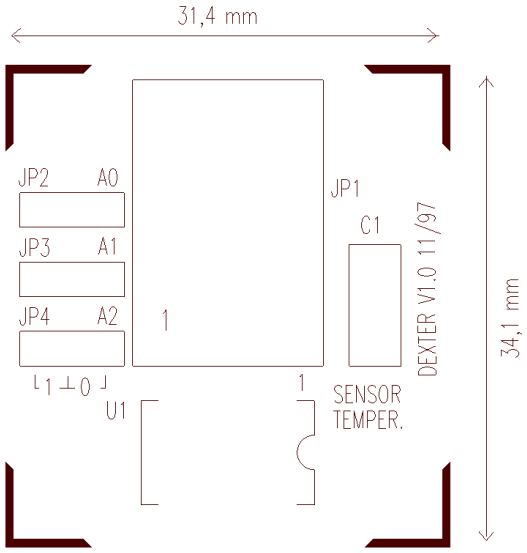
Ao ligar a alimentação elétrica o indicador luminoso de ENERGIA (LED) acende no painel do conversor A/D.

Os seguintes itens acompanham o equipamento:

- Conversor Analógico/Digital para µDX Série 100
- Cabo para conexão à rede DXNET
- Manual de Utilização (este manual)
- Fonte de Alimentação
- Sensor de Temperatura
- Cabo para conexão do sensor de temperatura
- Conexão dos Sensores de Temperatura

Para conexão dos sensores de temperatura existe um conector tipo RJ11 no Conversor Analógico/Digital. Acompanha o equipamento um cabo de 2 metros e um sensor de temperatura.

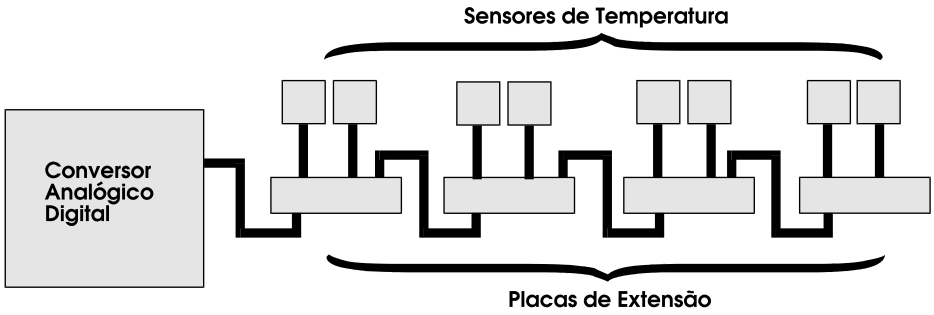
Para conectar sensores adicionais a DEXTER comercializa uma placa de extensão com 3 derivações e sensores adicionais. Note que todos os sensores utilizam a mesma linha de comunicação a 4 fios. O que os distingue é o endereço programado via estrapes (jumpers) na placa de cada sensor - 3 jumpers, permitindo endereço de 1 a 8.



Acima temos o layout da placa de sensor de temperatura . Note o conector RJ11 central (JP1), que conecta a placa ao conversor A/D. A esquerda existem 3 jumpers (JP2, JP3, JP4), responsáveis pelo endereço do sensor de temperatura. JP2 é o bit menos significativo deste endereço e JP4 o mais significativo. Se o jumper estiver ligado entre o pino à esquerda e o pino central o jumper estará em 1 (um). Já se o jumper estiver entre o pino central e o pino à direita o jumper estará em 0 (zero). A tabela a seguir indica o valor de JP2, JP3 e JP4 para os 8 endereços possíveis para o sensor de temperatura:

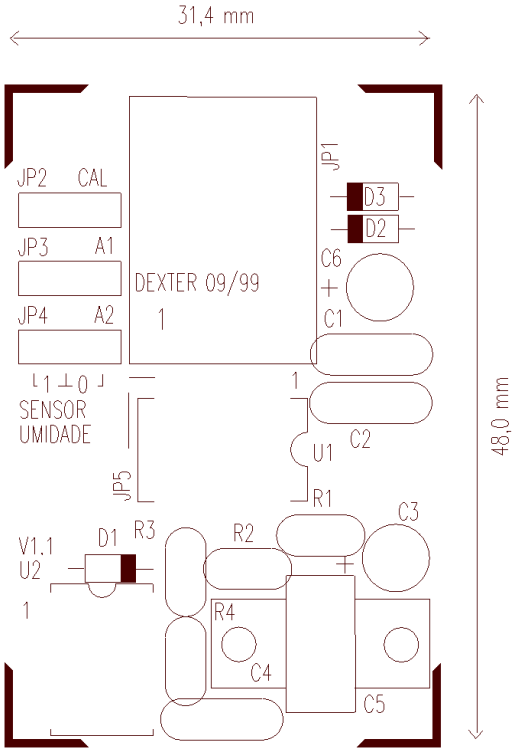
	JP4	**JP3**	**JP2**
Sensor de Temperatura 1	0	0	0
Sensor de Temperatura 2	0	0	1
Sensor de Temperatura 3	0	1	0
Sensor de Temperatura 4	0	1	1
Sensor de Temperatura 5	1	0	0
Sensor de Temperatura 6	1	0	1
Sensor de Temperatura 7	1	1	0
Sensor de Temperatura 8	1	1	1

A placa de extensão opcional permite derivar 3 ligações para sensores de temperatura a partir do cabo conectado ao conversor A/D. Assim, para conectar todos os 8 sensores de temperatura, por exemplo, são necessárias 4 placas de extensão (sendo que uma saída de uma das placas fica vazia):



Conexão dos Sensores de Umidade

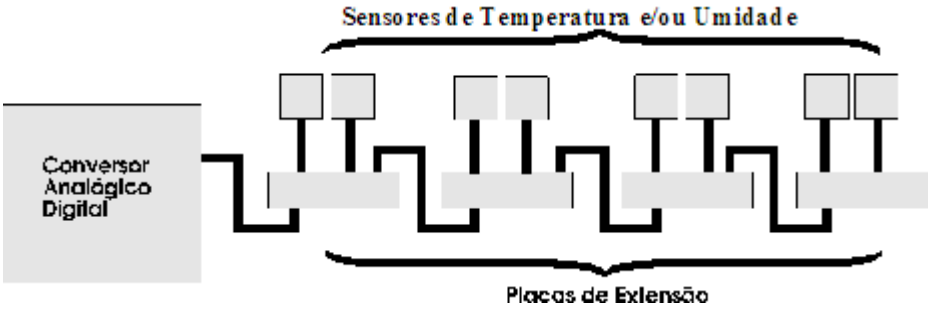
Os sensores de umidade são conectados no mesmo conector RJ11 dos sensores de temperatura. Para conectar vários sensores a DEXTER comercializa uma placa de extensão com 3 derivações. Note que todos os sensores utilizam a mesma linha de comunicação a 4 fios. O que os distingue é o endereço programado via estrapes (jumpers) na placa de cada sensor - 2 jumpers, no caso do sensor de umidade, permitindo endereço de 5 a 8. Ou seja, no caso do sensor de umidade apenas os conectores JP3 e JP4 possuem estrapes para programação do endereço do sensor.



Acima temos o layout da placa de sensor de umidade. Note o conector RJ11 central (JP1), que conecta a placa ao conversor A/D. A esquerda existem 2 jumpers (JP3, JP4), responsáveis pelo endereço do sensor de umidade. JP3 é o bit mais significativo deste endereço e JP4 o menos significativo. Se o jumper estiver ligado entre o pino à esquerda e o pino central o jumper estará em 1 (um). Já se o jumper estiver entre o pino central e o pino à direita o jumper estará em 0 (zero). A tabela a seguir indica o valor de JP3 e JP4 para os 4 endereços possíveis para o sensor de umidade:

	JP3	JP4
Sensor de Umidade 5	0	0
Sensor de Umidade 6	0	1
Sensor de Umidade 7	1	0
Sensor de Umidade 8	1	1

Note que os endereços de 1 a 4 do Conversor A/D não admitem sensores de umidade, devendo necessariamente ser utilizados apenas com sensores de temperatura. Além disso, um endereço usado como sensor de umidade não pode possuir sensor de temperatura e vice-versa, ou seja, eles são mutuamente excludentes. Por fim, o conector designado como CAL na placa impressa do sensor de umidade é usado apenas na fábrica para calibrar o sensor, e não deve ser conectado nenhum estrape nele. A placa de extensão opcional permite derivar 3 ligações para sensores de temperatura e/ou umidade a partir do cabo conectado ao conversor A/D. Assim, para conectar 4 sensores de temperatura + 4 sensores de umidade, por exemplo, são necessárias 4 placas de extensão (sendo que uma saída de uma das placas fica vazia):

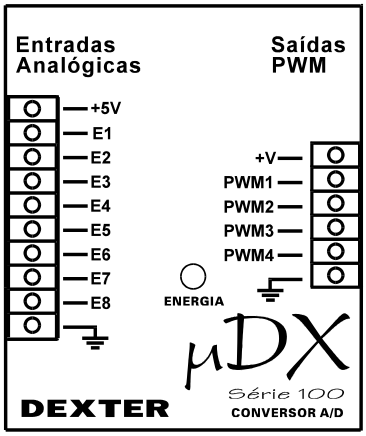


Conexão das Entradas Analógicas e Saídas PWM

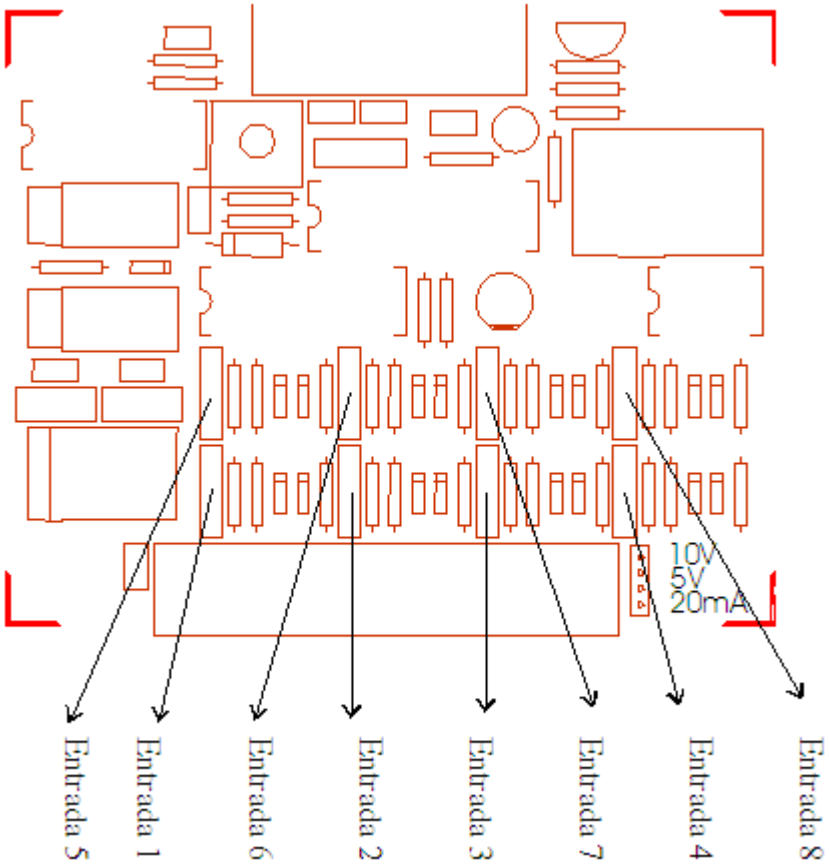
As entradas analógicas são disponíveis através de um conector de 10 terminais. Oito terminais são para as 8 entradas, um terminal é a referência (GND) e outro terminal é ligado a uma fonte regulada precisa de +5 Vdc. Abaixo temos o desenho do Conversor A/D com os terminais para as entradas analógicas e os terminais para as saídas PWM. As saídas PWM estão em um conector de 6 terminais, pois além das saídas existe um terminal de referência (GND) e um terminal com +V disponível (+12 Vdc).

Atenção: Nunca curto-circuite as saídas PWM com o terminal +V. Isto irá danificar a saída PWM e/ou todo o Conversor A/D.

A saída +V está ligada diretamente à entrada de alimentação do conversor A/D (+12 Vdc). Ela pode ser usada para alimentar algum circuito adicional para amplificar ou conformar o sinal de sensores ligados as entradas analógicas, por exemplo. A corrente máxima disponível neste terminal é de 150 mA.



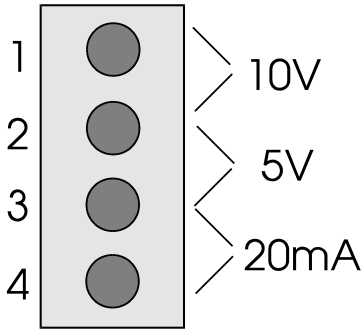
O conversor A/D sai de fábrica com todas as entradas analógicas programadas via jumper interno para a faixa de 0 a 10 Vdc. Para modificar esta programação é necessário abrir o equipamento. Para isso desconecte os cabos de DXNET e alimentação elétrica do Conversor A/D e abra a caixa metálica forçando levemente as laterais para afastarem-se dos encaixes que prendem a tampa ao fundo. Puxe cuidadosamente a tampa para cima. Abaixo temos um desenho da placa impressa do Conversor A/D:



Os jumpers correspondentes a cada uma das entradas analógicas são os marcados no desenho. Cada um destes jumpers permite 3 posições distintas, conforme o desenho no canto inferior direito da placa de circuito impresso.

Assim, ligando o jumper entre os pinos 1 e 2 a entrada fica programada para uma faixa de tensão entre 0 a 10 V. Já conectando o jumper entre os pinos 2 e 3 a entrada fica com faixa de tensão

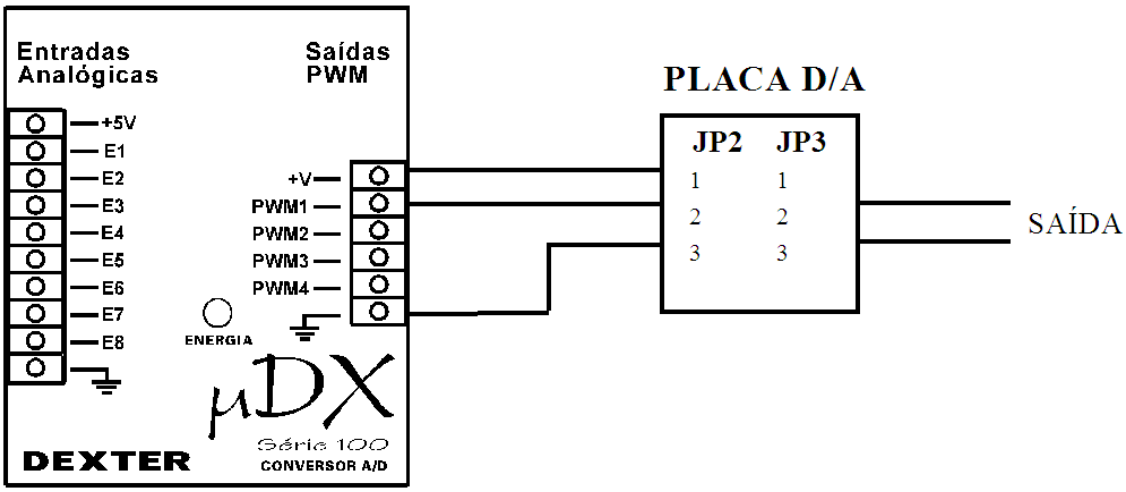
entre 0 e 5 V. Se conectarmos o jumper entre os pinos 3 e 4 a entrada será programada com faixa de corrente entre 0 e 20 mA. Por fim, retirando o jumper a entrada será para tensão entre 0 e 5 V com alta resistência de entrada (necessário no caso de fonte de sinal de alta impedância, como potenciômetros, por exemplo).



Conexão da Placa de Saída D/A

As saídas PWM (PWM1 a PWM4) do Conversor A/D permitem implementar saídas analógicas com o uso da Placa de Saída Digital/Analógica. Esta placa converte o sinal modulado em largura de pulso em um sinal analógico de 0 a 10V ou de 0 a 5V. A Placa de Saída D/A possui um estrape (jumper) que, quando aberto, força a saída a ser 0-10V. Já com este estrape fechado a saída se situa entre 0-5V. As conexões da placa de Saída D/A e a saída PWM1 do Conversor A/D são mostradas abaixo, assim como a função de cada pino dos conectores da placa D/A:

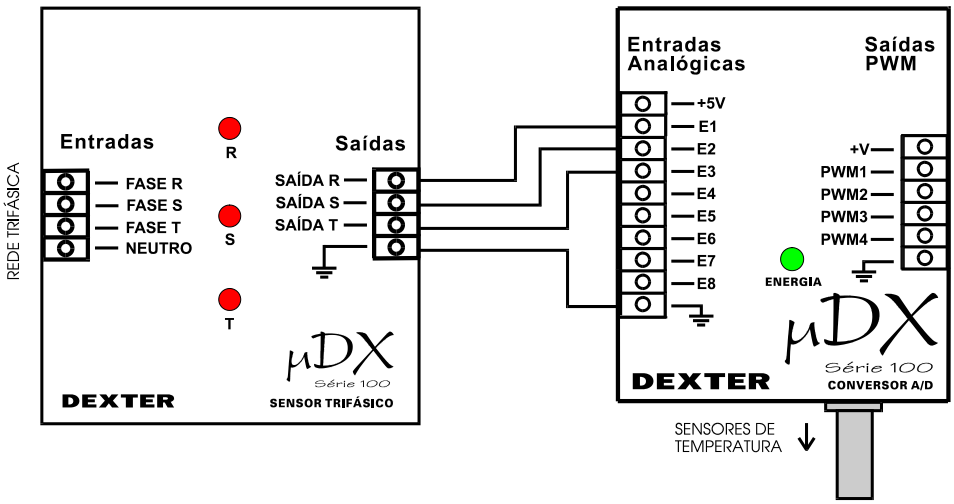
JP2	**Pino 1:**	Alimentação de 12Vdc. Ligar no pino +V do conector “Saídas PWM”.
	Pino 2:	Entrada PWM. Usar um dos pinos “PWM1” a “PWM4” do conector “Saídas PWM”.
	Pino 3:	Referência GND. Ligar no pino de GND do conector “Saídas PWM”.
JP3	**Pino 1:**	Saída de 12Vdc.
	Pino 2:	Saída de tensão analógica 0-5V (JP1 fechado) ou 0-10V (JP1 aberto).
	Pino 3:	Referência GND.



Conexão do Sensor Trifásico

O módulo de Sensor Trifásico permite, em conjunto com o Conversor A/D, monitorar a tensão de três fases da rede elétrica, nos limites entre 0 e 255 VCA. O Sensor Trifásico é constituído de três transformadores (construídos de forma a manterem linearidade para uma ampla faixa de tensões primárias), com saidas contínuas de 0 a 10V. Assim, as entradas do Conversor A/D utilizadas com o Sensor Trifásico devem ser programadas (via jumpers) para 0-10V. Abaixo é ilustrada a conexão entre os equipamentos:

Conversor A/D e Sensor Trifásico



Especificações Técnicas

- 8 entradas analógicas programáveis via jumper para 0 a 5V, 0 a 10V ou 0 a 20mA. Resolução de 8 bits. Entradas protegidas contra sobre-tensão. Resistência de entrada: 5KΩ ou 1MΩ para faixa 0 a 5V; 10KΩ para faixa 0 a 10V; 250Ω para faixa 0 a 20mA. Tempos de conversão programáveis entre 40ms e 320ms.
- Comunicação com até 14 µDXs via rede DXNET.
- 4 saídas analógicas (0 a 5V ou 0 a 10V) via Placa de Saída Analógica.
- 4 saídas por modulação de largura de pulso (PWM), compatíveis com o bloco PWM do controlador µDX (µDX com versão de firmware igual ou maior que 5.7).
- Sensor de temperatura digital, com resolução até 0,02°C e precisão de 0,5°C. Endereçamento do sensor via 3 jumpers, permitindo a conexão de até 8 sensores de temperatura ao Conversor A/D.
- Faixa de operação do sensor de temperatura: -55°C a 125°C.
- Sensor de umidade digital (opcional), com resolução até 0,5% UR e precisão de 5% UR. Endereçamento do sensor via 2 jumpers, permitindo a conexão de até 4 sensores de umidade ao Conversor A/D.
- Faixa de operação do sensor de umidade: 10% UR a 90% UR.
- Alimentação: 9-14 VDC @ 80mA (máximo).
- Temperatura de operação: 0°C até 60°C.

Modem

Este documento descreve as características do modem (modulador/demodulador) do µDX100. Com este modem, é possível estender a rede DXNET para qualquer ponto do planeta, através de linha telefônica discada ou privativa. Todas as instruções permitidas na DXNET são disponíveis remotamente via modem.

O modem tem capacidade de armazenar até 16 números telefônicos (em memória não volátil - E²PROM) para discagem e pode também detectar sinal de chamada (ring) e atender ao enésimo toque. A taxa de comunicação é de 300 bps, suficiente para os pequenos pacotes de comunicação que circulam na rede local do µDX100 (DXNET). O modem é compatível com norma Bell 103. O modem possui senha para habilitar o acesso remoto.

O equipamento possui uma saída RS-232C (apenas TX,RX e GND) e, opcionalmente, saída RS-485. A taxa de comunicação pode ser de 300, 600, 1200, 2400, 4800 e 9600 bps.

Modems com versão 2.6 ou superior podem ser operados, opcionalmente, como rádio-modems, acoplados a rádio-transmissor (modelo TK-760 Kenwood).

Conexão ao µDX

Acompanha o modem um cabo blindado, com pinos P2 nas extremidades, para conectá-lo à rede local DXNET. No modem existem dois conectores P2 fêmea (designados como DXNET na caixa), interconectados em paralelo. Basta conectar o cabo em qualquer um dos conectores DXNET do modem e a outra extremidade ao controlador programável µDX100. Além disso, é fornecido um cabo com conectores RJ11 nas extremidades para conexão do modem à rede telefônica e uma fonte de alimentação. Portanto, os itens que compõem o modem são:

- Modem para µDX Série 100
- Cabo para conexão à rede DXNET
- Cabo para conexão à rede telefônica
- Manual de Utilização (este manual)
- Fonte de Alimentação

Para conectar o modem à rede telefônica utilize o conector RJ11 fêmea designado como LINHA na caixa do modem. O outro conector (designado como FONE) serve para ligar um telefone a mesma linha.

A fonte de alimentação é que supre de energia o modem. A extremidade livre do cabinho que sai da fonte de alimentação deve ser ligada ao conector no modem que está indicado como ENERGIA.

Atenção: Antes de ligar a fonte de alimentação na rede elétrica verifique se ela está ajustada adequadamente para a tensão da rede no local (127 ou 220 VCA).

Ao ligar a alimentação elétrica o indicador luminoso de ENERGIA (led verde) acende no painel do modem. O outro indicador, CONEXÃO (led vermelho), pisca momentaneamente. Este led indica sinal de portadora (carrier) presente e é normal que pisque ao energizar o equipamento.

Modos de Operação

O modem tem 11 modos de operação, determinados pela variável v0 (variável interna de 8 bits do modem):

Variável v0	
xxxx0000	Modem desativado (linha telefônica desconectada). No caso de falta de energia ou reset continua neste estado.
nnnn0001	Atende ao enésimo chamado, ativando modem como destino. No caso de falta de energia ou reset derruba ligação corrente mas volta a este estado. Necessita de senha em v2 para ativar comunicações.
xxxx0010	Força modem ativo, como destino. No caso de falta de energia ou reset continua neste estado.
xxxx0011	Força modem ativo, como origem. No caso de falta de energia ou reset continua neste estado.
nnnn0100	Disca enésimo número telefônico, a seguir ativa modem como origem. No caso de falta de energia ou reset vai para estado de modem desativado.
nnnn0101	Força modem ativo, como destino, no modo de rádio-transmissor. No caso de falta de energia ou reset continua neste estado. Operacional a partir de modem versão 2.6 ou superior. Os 4 bits superiores (nnnn) indicam o atraso entre acionar o transmissor (PTT) e efetivamente transmitir o dado, em décimos de segundo (pode-se programar este tempo entre 0,1s e 1,6s).
nnnn0110	Força modem ativo, como origem, no modo de rádio-transmissor. No caso de falta de energia ou reset continua neste estado. Operacional a partir de modem versão 2.6 ou superior. Os 4 bits superiores (nnnn) indicam o atraso entre acionar o transmissor (PTT) e efetivamente transmitir o dado, em décimos de segundo (pode-se programar este tempo entre 0,1s e 1,6s).
nnnn0111	Força modem como interface entre microcomputador e rádio-modem. No caso de falta de energia ou reset continua neste estado. Operacional a partir de modem versão 2.6 ou superior. Os 4 bits superiores (nnnn) indicam o atraso entre acionar o transmissor (PTT) e efetivamente transmitir o dado, em décimos de segundo (pode-se programar este tempo entre 0,1s e 1,6s).
xxxx1000	Força discagem via modem externo conectado a porta serial. Este estado transmite o string "ATDn", sendo n o enésimo número telefônico gravado no modem. Ao terminar a transmissão do comando "ATD" o Modem para µDX retorna ao estado 0. Operacional a partir de modem versão 3.4 ou superior. Os 4 bits superiores (nnnn) indicam qual o número telefônico dos 16 armazenados no Modem deve ser usado. Note que o string de discagem "ATD" já está pré-gravado no Modem para µDX. Assim, não é necessário usar um dos 8 strings programáveis disponíveis. O Modem para µDX

aguarda o caracter "1" (de 1,10,11,12,13...) ou caracter "C" (de connect) para entender que a conexão via modem externo foi bem sucedida.

0nnn1001 Envia mensagem n para modem externo conectado a porta serial. Este estado transmite o string n (de 0 a 7) gravado na memória não-volátil do Modem para µDX. Ao terminar a transmissão o modem retorna ao estado 0. Operacional a partir de modem versão 3.4 ou superior. Os 3 bits superiores (0nnn) indicam qual dos strings pré-gravados será transmitido. O Modem para µDX aguarda o caracter "0" ou "O" (de OK) para saber que o modem externo entendeu o string transmitido.

xxxx0110 Força desconexão de modem externo ligado a porta serial. Ao comutar para este modo, o Modem para µDX transmite via serial o string "+++", de forma a retornar o modem ligado a serial para modo comando, e a seguir transmite "ATH" para encerrar a conexão. Operacional a partir de modem versão 3.4 ou superior. O Modem para µDX aguarda o caracter "0" ou "O" (de OK) para saber que o comando de desconexão foi efetuado.

Onde x indica bit irrelevante (tanto faz seu valor), e os valores expressos acima para v0 estão em binário. A variável v0 permite leitura e escrita. Note que apenas os 4 bits inferiores (nibble LSB) da variável v0 determinam o comportamento do modem. Os 4 bits superiores (nibble MSB) indicam o número de toques (rings) para atendimento, o número telefônico a ser discado (entre os 16 armazenados na memória não volátil do modem), ou ainda o atraso entre acionamento do rádio-transmissor e transmissão (em décimos de segundo).

Para comandar o modem, o controlador µDX ligado a ele deve escrever na variável v0 do modem. Para isso basta usar o bloco DXNET para comunicação entre µDXs. Por exemplo, para liberar a linha telefônica e manter o modem desativado basta zerar os 4 bits inferiores de v0 do modem.

Estado 0 - Modem Desativado

Valor de v0 do modem = xxxx0000 em binário.

Neste estado o modem fica desligado da linha telefônica (que pode ser usada normalmente). Apenas a comunicação serial via RS-232C ou RS-485 fica operacional.

No caso de uma reinicialização o modem mantém este estado. O led vermelho de conexão (carrier) do painel do modem permanece desligado.

Estado 1 - Atende ao Enésimo Chamado

Valor de v0 do modem = nnnn0001 em binário.

Este é o estado para que o modem trabalhe com linha discada, atendendo chamadas telefônicas após um número programável de toques (rings). O número de toques da campainha do telefone é programável pelos 4 bits superiores (nibble MSB) de v0, entre os valores de 1 toque (v0=00000001) até o máximo de 16 toques (v0=11110000). Após receber o número de toques programado, o modem é ativado como destino e a linha telefônica é ocupada.

Na comunicação entre modems é necessário que um deles seja programado como origem e o outro como destino, para que suas freqüências de portadoras não sejam coincidentes. O modem origem é o que gerou a chamada e o modem destino é o que recebe uma chamada telefônica.

Ao atender a chamada, caso do outro lado exista um modem transmitindo sua portadora, o indicador luminoso (led vermelho) chamado de conexão no painel é ativado.

Todas as comunicações serão bloqueadas até que o modem destino receba a senha correta na sua variável v2. Caso ele receba uma senha incorreta irá imediatamente derrubar a ligação telefônica. Além disso, começa a contar tempo para derrubar a linha por "time-out". Se após 2 minutos ainda não tiver recebido a senha correta a ligação também é interrompida.

Após receber a senha correta, o modem destino libera comunicações, e a partir daí se tem total acesso a rede local DXNET conectada a este modem. No caso de um µDX ter originado a ligação, ele pode transmitir o estado de nodos ou o valor de variáveis aos µDXs remotos. Já no caso de existir um microcomputador na rede local DXNET do modem que originou a chamada telefônica, este pode acessar todas as funções dos µDXs remotos (ligados ao modem destino, que atendeu a chamada telefônica), inclusive transmitir um novo programa ao controlador µDX100 remoto.

Durante toda chamada telefônica, os modems ficam atentos a erros de comunicação. Se os erros se prolongarem por 2 minutos, a ligação é interrompida por "time-out". Isto evita que a linha telefônica fique "pendurada" (ocupada indefinidamente) no caso de algum erro catastrófico.

No caso de falta de energia elétrica ou reinicialização (reset) o modem continua neste estado, mas derruba a ligação telefônica corrente. Note que o estado do modem (v0) é armazenado em memória não volátil (E²PROM).

Estado 2 - Força Modem Ativo como Destino

Valor de v0 do modem = xxxx0010 em binário.

Neste estado a linha telefônica é ocupada pelo modem e este é programado como destino. Este estado, junto com o seguinte (força modem ativo, como origem), permite utilizar uma linha telefônica privativa. Este estado se mantém mesmo após uma reinicialização ou falta de energia elétrica. Também não importa os erros que venha a ocorrer na comunicação, o estado do modem não se altera (não existe "time-out", como no estado anterior).

Este estado não requer senha para iniciar o acesso.

Estado 3 - Força Modem Ativo, como Origem

Valor de v0 do modem = xxxx0011 em binário.

Este é o estado complementar do estado anterior. Caso os modems estejam conectados por linha telefônica privativa pode-se simplesmente mante-los conectados programando-se um como origem (estado 3) e o outro como destino (estado 2).

Os modems estarão constantemente conectados.

Estado 4 - Disca Enésimo Número Telefônico

Valor de v0 do modem = nnnn0100 em binário.

Este estado é complementar ao estado 1 (atende ao enésimo toque). Ele permite escolher um número telefônico entre os 16 programáveis na memória não volátil do modem. Após discar, o modem comuta automaticamente para estado 2 - Força modem ativo, como origem. A discagem é feita por pulsos (relação de ½ entre tempo de linha com carga e linha livre).

Ao ser atendida a chamada, caso do outro lado exista um modem transmitindo sua portadora, o indicador luminoso (led vermelho) chamado de conexão no painel é ativado.

Todas as comunicações serão bloqueadas até que o modem destino receba a senha correta na sua variável v2. Esta providência deve ser tomada pelo µDX que fez a requisição de chamada telefônica para o modem origem. Da mesma forma que o estado 1, este estado começa a contar tempo para derrubar a linha por "time-out". Se após 2 minutos não for estabelecida comunicação a ligação é interrompida.

No caso de interrupção de rede elétrica ou reset, o modem retorna ao estado 0, desativando o modem e liberando a linha telefônica.

Estado 5 – Força Modem Ativo, como Destino, para Rádio-Modem

Valor de v0 do modem = nnnn0101 em binário.

Neste estado o modem (versão 2.6 ou superior) é programado como destino, via rádio-transmissão. Este estado, junto com o seguinte (força modem ativo, como origem, para rádio-modem), permite utilizar um rádio-transmissor (Kenwood TK-760, ou Motorola PRO3100) para comunicação de dados. Este estado se mantém mesmo após uma reinicialização ou falta de energia elétrica. Também não importa os erros que venha a ocorrer na comunicação, o estado do modem não se altera (não existe "time-out", como no estado anterior).

Este estado não requer senha para iniciar o acesso. Os 4 bits superiores do valor de v0 (nnnn) indicam o atraso entre acionar o transmissor (PTT) e efetivamente transmitir o dado, em décimos de segundo (pode-se programar este tempo entre 0,1s e 1,6s). Este atraso deve ser ajustado conforme o rádio-transmissor utilizado. Aconselha-se iniciar com um valor alto e diminuir-se paulatinamente, até obter-se o limite de velocidade do sistema.

Estado 6 - Força Modem Ativo, como Origem, para Rádio-Modem

Valor de v0 do modem = nnnn0110 em binário.

Este é o estado complementar do estado anterior. Caso os modems (versão 2.6 ou superior) estejam conectados por rádio-transmissor (Kenwood TK-760 ou Motorola PRO3100) pode-se simplesmente mantê-los conectados programando-se um como origem (estado 6) e o outro como destino (estado 5).

Os rádio-modems estarão constantemente conectados, embora a transmissão só seja ativada quando houver necessidade. A comunicação completa de um pacote de dados leva cerca de 0,8 segundos (dependendo dos atrasos programados e do tipo de rádio-transmissor usado).

Os 4 bits superiores do valor de v0 (nnnn) indicam o atraso entre acionar o transmissor (PTT) e efetivamente transmitir o dado, em décimos de segundo (pode-se programar este tempo entre 0,1s e 1,6s). Este atraso deve ser ajustado conforme o rádio-transmissor utilizado. Aconselha-se iniciar com um valor alto e diminuir-se paulatinamente, até obter-se o limite de velocidade do sistema.

Estado 7 - Força Modem como Interface entre Microcomputador e Rádio-modem

Valor de v0 do modem = nnnn0111 em binário.

Este estado permite interfacear um microcomputador IBM-PC compatível com o rádio-modem. Na verdade, no estado 7 o modem converte as mensagens recebidas via porta serial em mensagens DXNET adequadas aos tempos de transação do rádio-modem. Nas últimas páginas deste manual existe um diagrama, exemplificando a conexão por rádio-transmissão, entre uma estação central (microcomputador com software supervisório PGR ou Elipse) e três estações remotas.

Os 4 bits superiores do valor de v0 (nnnn) indicam o atraso entre acionar o transmissor (PTT) e efetivamente transmitir o dado, em décimos de segundo (pode-se programar este tempo entre 0,1s e 1,6s). Este atraso deve ser programado igual ao utilizado nos modems ligados aos rádio-transmissores (estados 5 e 6).

Estado 8 - Força conexão via Modem Externo

Valor de v0 do modem = nnnn1000 em binário.

Este estado permite utilizar um modem externo ligado a porta serial do Modem para µDX100 para efetuar uma discagem. Este estado transmite o string "ATDn", sendo n o enésimo número telefônico gravado no modem. Ao terminar a transmissão do comando "ATD" o Modem para µDX

retorna ao estado 0. Operacional a partir de modem versão 3.4 ou superior. Os 4 bits superiores (nnnn) indicam qual o número telefônico dos 16 armazenados no Modem deve ser usado. Note que o string de discagem "ATD" já está pré-gravado no Modem para µDX. Assim, não é necessário usar um dos 8 strings programáveis disponíveis. O Modem para µDX aguarda o caracter "1" (de 1,10,11,12,13...) ou caracter "C" (de connect) para entender que a conexão via modem externo foi bem sucedida. Com isso, tanto se o modem externo estiver no modo verbal quanto no modo de retorno numérico o Modem para µDX irá entender o retorno do comando ATD. Caso a conexão não seja bem sucedida o Modem para µDX irá ligar o bit 4 da variável v1 (status do Modem), indicando erro. Nas últimas páginas deste manual existe um programa de exemplo que utiliza um telefone celular para conexão remota com µDX.

Estado 9 - Transmite string p/Modem Externo

Valor de v0 do modem = 0nnn1001 em binário.

Este estado permite comandar um modem externo ligado a porta serial do Modem para µDX. Envia mensagem n para modem externo conectado a porta serial. Este estado transmite o string n (de 0 a 7) gravado na memória não-volátil do Modem para µDX. Ao terminar a transmissão o modem retorna ao estado 0. Operacional a partir de modem versão 3.4 ou superior. Os 3 bits superiores (0nnn) indicam qual dos strings pré-gravados será transmitido. O Modem para µDX aguarda o caracter "0" ou "O" (de OK) para saber que o modem externo entendeu o string transmitido. Caso a transmissão de comandos AT não seja bem sucedida o Modem para µDX irá ligar o bit 4 da variável v1 (status do Modem), indicando erro. Nas últimas páginas deste manual existe um programa de exemplo que utiliza um telefone celular para conexão remota com µDX.

Estado 10 - Força desconexão via Modem Ext.

Valor de v0 do modem = xxxx1010 em binário.

Este estado permite desconectar um modem externo ligado a porta serial do Modem para µDX. Força desconexão de modem externo ligado a porta serial. Ao comutar para este modo, o Modem para µDX100 transmite via serial o string "+++", de forma a retornar o modem ligado a serial para modo cmando, e a seguir transmite "ATH" para encerrar a conexão. Operacional a partir de modem versão 3.4 ou superior. O Modem para µDX aguarda o caracter "0" ou "O" (de OK) para saber que o comando de desconexão foi efetuado. Caso a desconexão não seja bem sucedida o Modem para µDX irá ligar o bit 4 da variável v1 (status do Modem), indicando erro. Nas últimas páginas deste manual existe um programa de exemplo que utiliza um telefone celular para conexão remota com µDX.

Obs.: Para permitir que o modem externo aceite chamadas (ou seja, permita que uma central disque para o µDX remoto e efetue conexão) basta transmitir para o modem externo o string "ATS1=n", sendo n o número de rings para atendimento da chamada telefônica. Neste caso, convém habilitar a senha do Modem via serial, de forma que a conexão só seja permitida se a estação de origem da conexão telefônica transmita a senha correta para o Modem do µDX100.

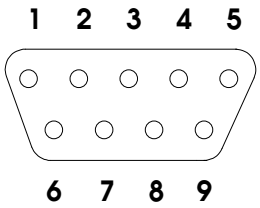
Interface Serial RS232C ou RS485

O modem possui também uma interface serial, padrão RS-232C, para comunicação por fio. No caso da RS-232C, são necessários 2 fios + malha (TX,RX,GND). A comunicação via serial, da mesma forma que via modem, forma uma extensão da rede DXNET. A taxa de transmissão é igual à do modem, ou seja, 300 bps, 8 bits, sem paridade, 1 stop bit. Como a taxa é lenta, o cabo de conexão pode ter até 500 metros. O sinal de saída varia entre 12V e -10V, aproximadamente.

Opcionalmente, pode-se instalar também uma interface RS-485, o que permite conexão via cabo por, pelo menos, alguns quilômetros. Neste caso, o fio pode ser um par trançado comum, utilizado para ligações telefônicas.

Por fim, modems de versão igual ou superior a 2.6 permitem conexão a rádio-transmissores Kenwood (modelo TK-760) ou Motorola (modelo PRO3100), controlando as linhas de TX, RX,

PTT e SQL. Veja diagrama de conexão nas páginas finais deste manual. Abaixo temos a pinagem do conector DB-9 macho utilizado para RS-232C e RS-485:



Conector DB-9 macho
Vista frontal

Pinagem:	Pino 1	TX (Rádio)
	Pino 2	RX (RS-232C)
	Pino 3	TX (RS-232C)
	Pino 4	RX (Rádio)
	Pino 5	GND (terra lógico)
	Pino 6	SQL (Rádio)
	Pino 7	PTT (Rádio)
	Pino 8	TX/RX B (RS-485)
	Pino 9	TX/RX A (RS-485)

Regulador Chaveado

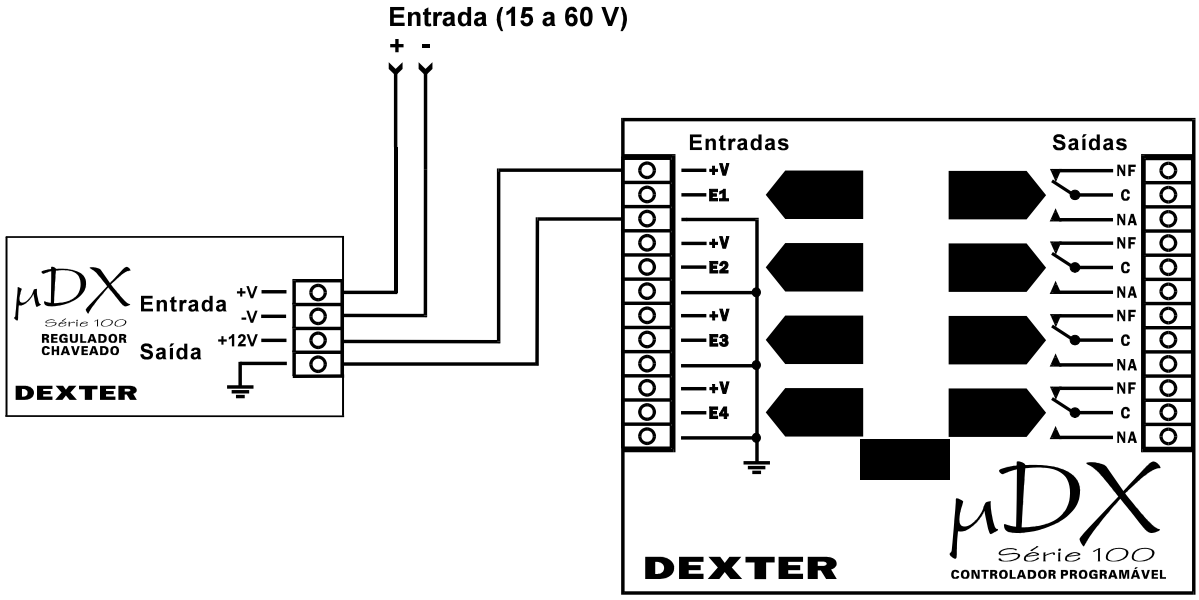
Este documento descreve as características do regulador chaveado para o controlador programável µDX100. O regulador permite alimentar um controlador µDX100, uma expansão de entradas/saídas e um modem para µDX100 (a corrente máxima fornecida é de 700mA, suficiente para todos estes equipamentos, mesmo com todos os 13 relés acionados - 4 do µDX + 8 da Expansão + 1 do Modem). Outras configurações podem ser estudadas através da soma das correntes máximas de cada dispositivo (ver tabela adiante, nas especificações técnicas). Esta soma não deve exceder o limite de 700mA.

A tensão de entrada deve estar entre 15 e 60V, sendo que a saída fornece 12V regulados @ 700mA. A saída é protegida contra curto-circuitos, sendo que o circuito limita a corrente de saída em cerca de 1,5A. Existe um fusível interno em série com a entrada de alimentação do regulador chaveado. Além disso, a entrada é protegida contra inversão de polaridade.

Conexão ao µDX100

Para conectar o regulador chaveado ao controlador programável µDX100 basta efetuar as ligações ilustradas no diagrama abaixo. Note que a saída +12V do regulador é ligada a uma das entradas +V do µDX100, enquanto a referência (terra) do regulador é conectado à mesma entrada no µDX. Note que o pino com o desenho de terra está ligado na referência da fonte de alimentação do µDX100. Este pino não deve, em hipótese alguma, ser aterrado à carcaça do equipamento controlado pelo µDX!

Outra possibilidade é utilizar a entrada de energia existente na lateral do µDX. Para isso, deve ser preparado um cabo com um lado livre (para conexão ao regulador via conector parafusado) e o outro lado com um conector tipo jack de força (para ligação ao µDX). Note que, no caso de alguns periféricos, como modem e Interface Homem/Máquina (IHM), este cabo é essencial, pois não existe possibilidade de conexão de força via fio parafusado. Cuidado para não inverter a polaridade no conector de força!



A entrada do regulador chaveado deve ser conectada a uma fonte de tensão contínua de 15 a 60 Vdc.

Fusível de entrada (interno):	1 A, rápido, 20 mm x 5 mm

A máxima corrente de saída é 700 mA, como enfatiza o quadro abaixo:

Corrente de saída:	0,7 A, para 15V ≤ Ventrada ≤ 60V

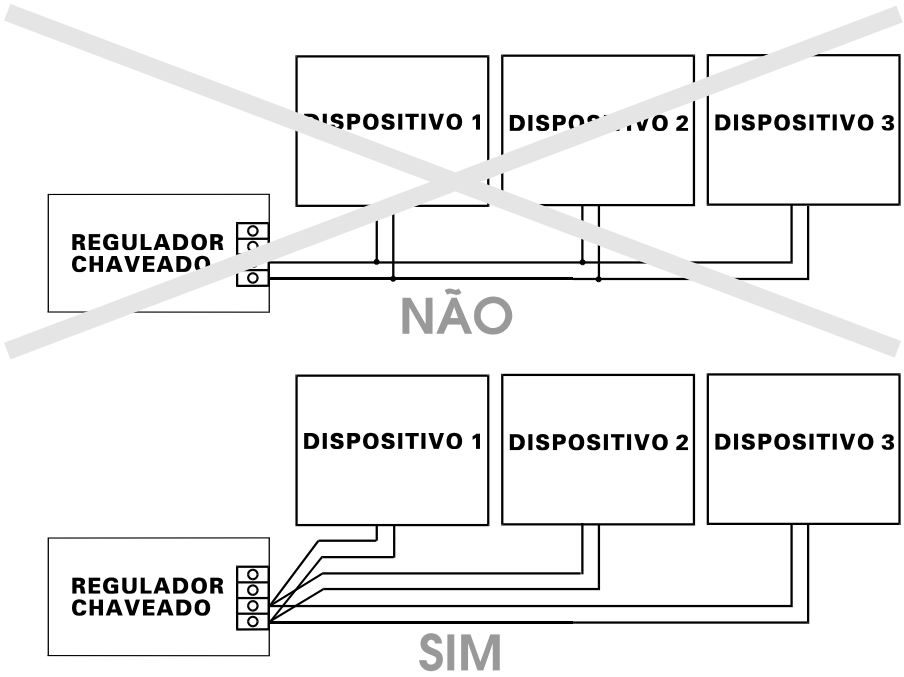
No caso de curto-circuito na saída, a corrente fica limitada em cerca de 1,5A. Antes de energizar o regulador chaveado, revise sua ligação de entrada e a conexão da saída do regulador aos equipamentos. Verifique cuidadosamente se a polaridade está correta.

ATENÇÃO: Não inverta a polaridade na entrada do regulador chaveado ou na entrada do µDX. No primeiro caso, o regulador simplesmente não ligará. Mas uma inversão na polaridade de alimentação do µDX o danificará.

No caso de alimentação de rede telefônica (-48V), ligue o terminal de -48V na entrada -V do regulador chaveado e o terra da alimentação de rede telefônica na entrada +V do regulador chaveado. Assim, do ponto de vista do regulador, ele estará sendo suprido com +48V. Entretanto, cuidado com ligações de terra do µDX100, pois o terra do µDX estará com -48V em relação à linha telefônica.

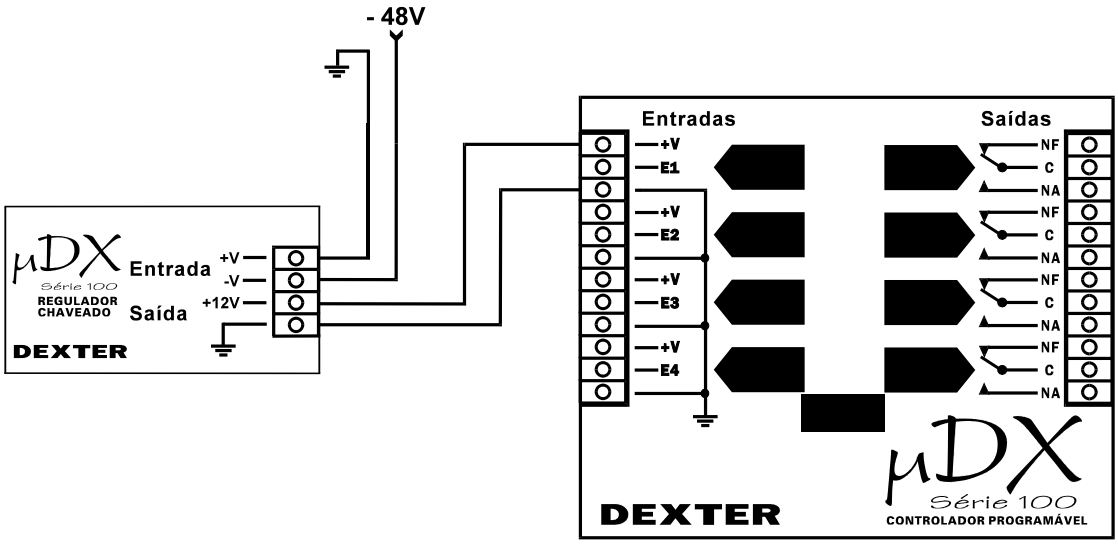
ATENÇÃO: Não existe isolação galvânica entre a entrada e saída do regulador chaveado.

Ao alimentar vários equipamentos com o mesmo regulador chaveado, use um cabo de alimentação para cada equipamento. Una estes cabos no conector de saída do regulador, como mostrado abaixo. Além disso, certifique-se que o consumo não ultrapassa 700 mA.



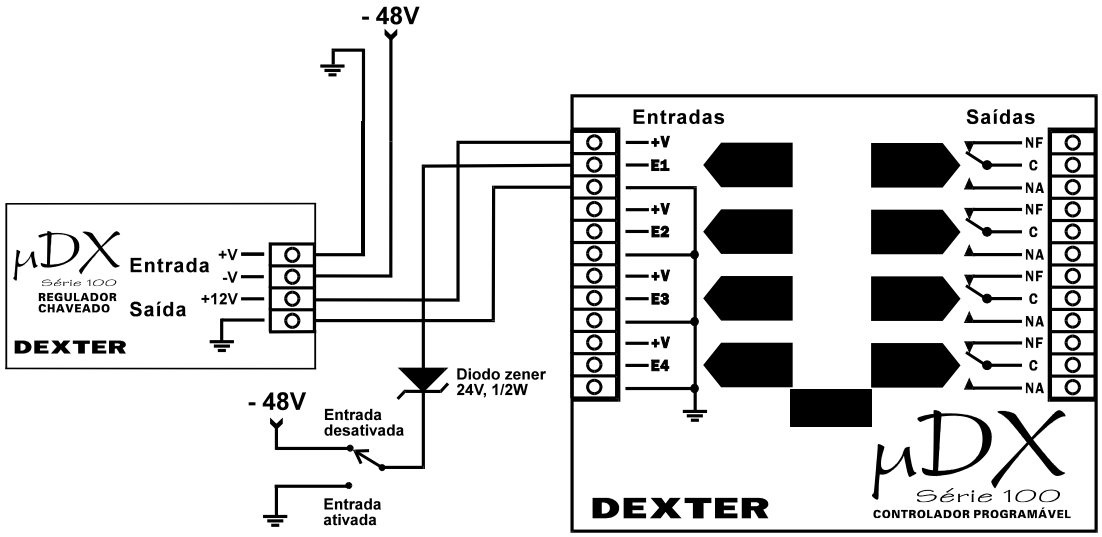
Conexão em linha de -48V

Em aplicações de telefonia, é comum a existência de alimentação contínua de tensão em -48V. Neste caso, deve-se usar o regulador chaveado como mostrado abaixo:



Ou seja, neste caso liga-se o terminal de -48V na entrada -V e terra na entrada +V. A excitação das entradas do µDX (que não são opto-acopladas) irá sofrer uma inversão lógica (facilmente contornável via software - bloco inversor do PG). Assim, ao aplicarmos -48V a uma das entradas E1, E2, E3 ou E4, esta entrada irá permanecer desativada (pois do ponto de vista do µDX os -48V são a referência). Já se aterrarmos (ligar à referência dos -48V) esta entrada ela se energizará (pois para o µDX estará sendo aplicada uma tensão de 48V positivos na entrada).Embora o µDX permita uma tensão de 48V nas entradas E1, E2, E3 ou E4, os limites de decisão não são adequados. De -48V a -47,1V a entrada estará desativada; e de -47,1V a 0V estará ativa. Para migrar este limiar de decisão, por exemplo, para o meio da escala (cerca de -24V), basta utilizar um diodo zener em série com a entrada (como mostrado na figura da próxima página).

Este diodo zener permite, virtualmente, ajustar o ponto de decisão como quisermos. No exemplo, utilizando um zener de 24V, teremos que de -48V a -24V (aproximadamente) a entrada do µDX estará desativada; e de -24V a 0V esta entrada ficará ativa.



Note que o limite de corrente do regulador chaveado é de 700mA. Além disso, muito cuidado para respeitar a polaridade de Ventrada, como ilustrado acima.

É possível conectar 1 Controlador µDX + 1 Expansão de Entradas/Saídas + 1 Modem em um

único Regulador Chaveado, alimentado com -48V, pois o consumo ficará dentro do limite citado acima.

Especificações Técnicas

Entrada:	15 a 60 Vdc.	
Saída:	12 Vdc 5% @ 700mA, para 15V Ventrada 60V.	
	Regulação de linha e carga melhor que 1%.	
	Saída protegida contra curto-circuito.	
	Corrente de curto-circuito de cerca de 1,5 A.	
	Desligamento automático caso temperatura excessiva.	
Rendimento:	65% (a plena carga, com Ventrada = 60V).	
Temperatura de Operação:	0°C até 40°C.	
Freqüência de Chaveamento:	50KHz.	
Consumo Máximo:	µDX	250mA
	Expansão	350mA
	IHM	80mA
	Modem	100mA
	Opto	0 mA
	Conversor A/D	20mA

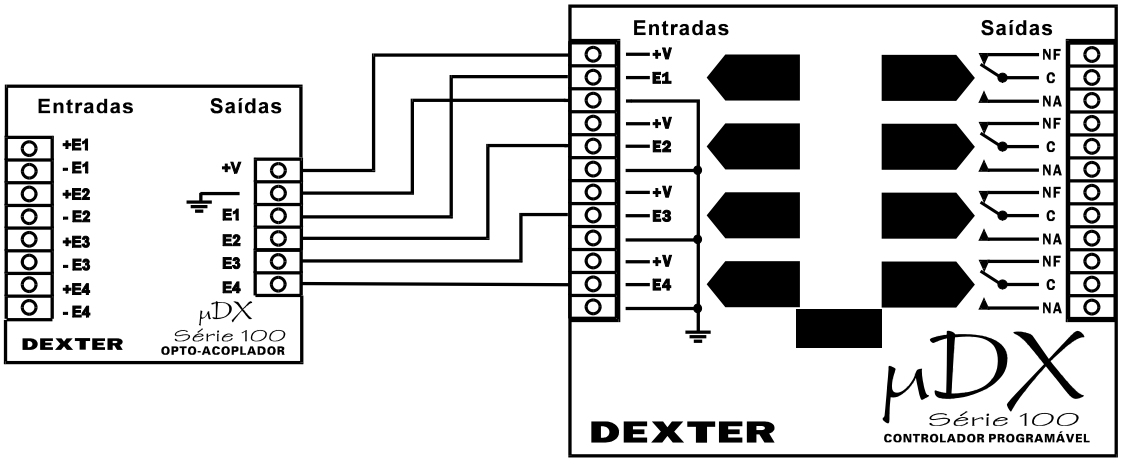
Observação: No caso do µDX e da expansão, cada relé não acionado diminui a corrente consumida em cerca de 40mA. Assim, se ligarmos simultaneamente, no máximo, 9 relés dos 12 disponíveis no µDX + expansão, o consumo deve ficar em cerca de: 250mA + 350mA - 120mA = 480mA. Ou seja, neste caso é possível alimentar outros circuitos, até o limite de 220mA (que somados aos 480mA perfazem 700mA).

Opto-acoplador

Este documento descreve as características do opto-acoplador para as entradas do µDX100 (4 entradas). Sendo opto-acopladas, as entradas permitem ligação direta a rede elétrica (127 ou 220 VAC) e oferecem alta imunidade contra ruídos. O opto-acoplador é conectado ao µDX100 através de fios. O suprimento de energia elétrica é retirado da fonte de alimentação do µDX100.

Conexão ao µDX

Para conectar o opto-acoplador ao controlador programável µDX100 basta efetuar as ligações ilustradas no diagrama abaixo. Note que as entradas E1 a E4 do µDX vão ligadas as correspondentes saídas E1 a E4 no opto-acoplador. Caso não se queira que determinada saída do µDX fique opto-acoplada, basta não efetuar a sua ligação ao opto-acoplador. Além das ligações de E1 a E4, basta conectar a fonte +V e o terra do µDX aos correspondentes pinos do opto-acoplador. Note que o pino com o desenho de terra está ligado na referência da fonte de alimentação do µDX. Este pino não deve, em hipótese alguma, ser aterrado à carcaça do equipamento controlado pelo µDX!



Uma vez conectado, o opto-acoplador já está apto a operar. Entretanto, é necessário programar as entradas (via jumpers internos) do opto-acoplador para o nível de tensão desejado.

Entradas

As entradas do opto-acoplador para µDX100 são todas opto-isoladas, oferecendo isolação galvânica. Assim, é possível conectá-las diretamente à rede elétrica (127 ou 220 VAC) ou usar várias fontes de sinal, sem conexão de referência (terra) entre elas.

A designação dada as entradas é E1 a E4.

Internamente ao opto-acoplador existem diversos "jumpers" removíveis, que permitem configurar individualmente as quatro entradas. Para acessá-los é necessário abrir a caixa do equipamento.

Abra a caixa do opto-acoplador forçando levemente as laterais para afastarem-se dos encaixes que prendem a tampa ao fundo. Puxe a tampa cuidadosamente para cima. Os "jumpers" internos permitem configurar as entradas do opto-acoplador para quatro tipos de sinal:

1) Alta Tensão AC	110 a 220 VAC (corrente alternada de 50 ou 60 Hz)
2) Alta Tensão DC	110 a 220 VDC (corrente contínua)
3) Baixa Tensão AC	6 a 30 VAC (corrente alternada de 50 ou 60 Hz)
4) Baixa Tensão DC	6 a 30 VDC (corrente contínua)

Atenção: Caso alguma entrada esteja preparada para a opção de baixa tensão (6 a 30 V), a conexão direta a rede elétrica (127 ou 220 VAC) provocará sua queima instantânea.

Normalmente o opto-acoplador é remetido com os "jumpers" instalados para a opção 1 (alta tensão AC), pois neste caso evita-se a queima acidental de uma das entradas ao ligá-la direto à rede elétrica.

Já a situação contrária, ou seja, ligar em 6 a 30 V uma entrada programada para 110 a 220 V, não traz maiores conseqüências (embora a entrada não consiga ser energizada por uma tensão tão baixa).

Na placa do opto-acoplador existem duas colunas de "jumpers", numerados de JP3 a JP6 e de JP7 a JP10. Estes jumpers permitem comutar o modo de operação das entradas. A coluna de jumpers JP3 a JP6 comuta entre alta tensão (jumper aberto) e baixa tensão (jumper fechado). Já a coluna JP7 a JP10 comuta entre corrente contínua (jumper aberto) e corrente alternada (jumper fechado). Assim, estando o jumper na coluna JP3 a JP6 fechado a entrada correspondente estará programada para baixa tensão. Com este jumper aberto a entrada será para alta tensão. Já se o jumper correspondente a mesma entrada estiver instalado na coluna JP7 a JP10 a entrada estará preparada para corrente alternada.

Note que uma entrada para corrente alternada (baixa ou alta tensão) pode ser acionada por uma corrente contínua sem nenhum problema (apenas ocorrerá um leve atraso para acionar ou desacionar a entrada (cerca de 50 ms) devido a filtragem). A recíproca não é verdadeira, ou seja, uma entrada programada para corrente contínua, ao ser excitada com corrente alternada, passará a ligar e desligar aleatoriamente (depende do ciclo de amostragem do µDX coincidir com o ciclo da rede elétrica).

A seguir temos uma tabela com a programação para todas as entradas do opto-acoplador (1 indica jumper fechado e 0 indica jumper aberto):

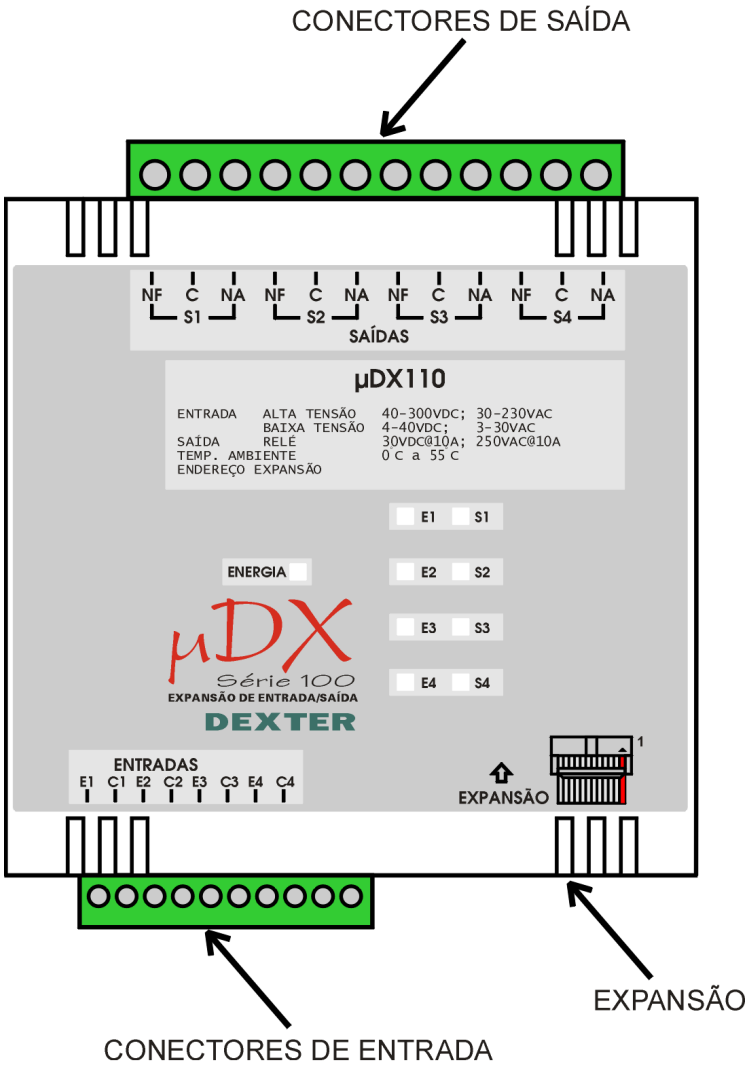
		Alta Tensão		Baixa Tensão	
Entrada	**Jumpers**	**DC**	**AC**	**DC**	**AC**
E1	JP3, JP7	00	01	10	11
E2	JP4, JP8	00	01	10	11
E3	JP5, JP9	00	01	10	11
E4	JP6, JP10	00	01	10	11

Por exemplo, se quisermos a entrada E2 em alta tensão AC devemos abrir o jumper JP4 e fechar o jumper JP8. Para programar E4 como baixa tensão AC devemos fechar JP6 e JP10 e assim por diante. Note que no caso de alta tensão pode-se ligar a entrada tanto em rede de 110 VCA quanto 220 VCA.

Atenção: No caso de entrada em corrente contínua deve ser respeitada a polaridade inscrita na tampa superior do equipamento. Já em corrente alternada (50 ou 60 Hz) a polaridade perde significado.

Expansão µDX110

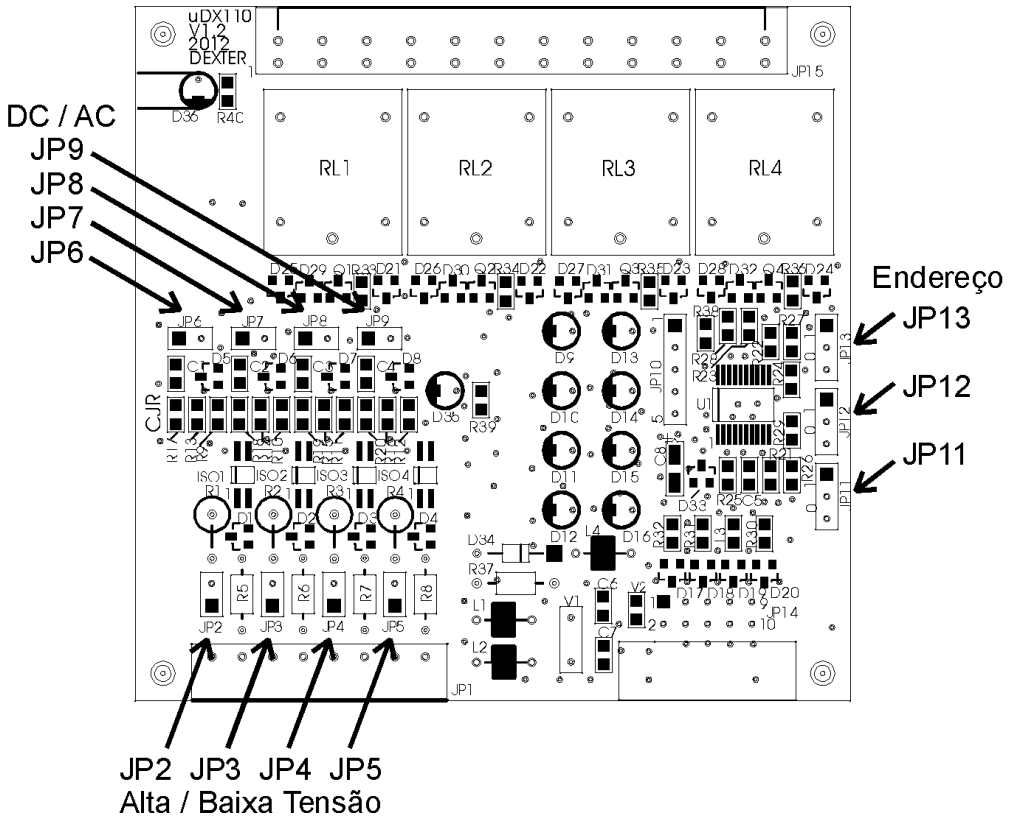
A Expansão µDX110 permite acrescentar quatro entradas opto-isoladas e quatro saídas à relé ao controlador µDX100 ou µDX101. Ela possui o mesmo dimensional do µDX101 e, ao contrário da Expansão de Entradas e Saídas para µDX100, sua conexão ao controlador é do tipo barramento, ou seja, todas são conectadas em paralelo. Para determinar o endereço da Expansão µDX110 existem 3 jumpers internos (até 8 Expansões µDX110 em um único controlador µDX100 ou µDX101).



Jumpers

A Expansão de Entradas/Saídas µDX110 está equipada com quatro saídas a relés eletromecânicos, para correntes de até 10 A.

A especificação das entradas depende dos jumpers internos. Os jumpers da Expansão µDX110 têm a localização e função especificadas abaixo. Note que os jumpers JP2 a JP9 se referem à programação de tensão das entradas do µDX110. Ainda existem três jumpers adicionais que permitem programar o endereço da Expansão (de módulo 1 até módulo 8). Para abrir a caixa metálica do µDX110 retire os dois parafusos (fenda cruzada) existentes nas laterais da caixa, e force levemente as laterais para que se afastem dos encaixes que a prendem ao fundo da caixa. Cuidado com os leds soldados a placa impressa, de forma a não danificá-los.



O endereço do µDX110 é determinado pelos jumpers existentes no jumpers JP11 até JP13, sendo que cada jumper pode ser posicionado como nível 0 ou nível 1. Note que o µDX110 é fornecido com os três jumpers de endereçamento em nível 0, ou seja, ocupando o endereço de módulo 1. Assim, temos:

Módulo	JP13	JP12	JP11
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

Lembre-se que todas as Expansões de Entrada/Saída µDX110 ligadas a um mesmo controlador µDX101 devem possuir endereços distintos.

Atenção: A alimentação elétrica da Expansão µDX110 advêm do Controlador µDX101. Assim, certifique-se de que a potência da fonte de alimentação do µDX101 é compatível com a quantidade de Expansões a serem conectadas ao mesmo. A fonte que é fornecida com o µDX101 pode suportar, no máximo, duas Expansões µDX110.

Atenção: A Expansão µDX110 pode ser usada também no µDX100, assim como a Expansão para µDX100 pode ser usada no µDX101. Entretanto, ao contrário da Expansão para µDX100, o µDX110 utiliza uma ligação em barramento e exige que seja utilizado o cabo paralelo (flat-cable) compatível com esta conexão. Nunca use o cabo que acompanha a Expansão para µDX100 com a Expansão µDX110, ou vice-versa, sob pena de danificar os equipamentos. Também não é possível mesclar as Expansões: um controlador µDX100 ou µDX101 só suporta tipos idênticos de Expansão em seu conector de expansão.

Já os jumpers JP2 a JP9 programam a faixa de tensão das entradas do µDX110, além do tipo de sinal (corrente contínua ou alternada). Note que as entradas são todas optoisoladas, oferecendo isolação galvânica. Assim, é possível conectá-las diretamente à rede elétrica (127 ou 220Vac) ou usar várias fontes de sinal, sem conexão de referência (terra) entre elas.

Os jumpers internos permitem configurar as entradas do µDX110 para quatro tipos de sinal:

Alta tensão AC **80 a 230 Vac (corrente alternada de 50 ou 60Hz)**

Alta tensão DC **80 a 230 Vdc (corrente contínua)**

Baixa tensão AC **4 a 30 Vac (corrente alternada de 50 ou 60Hz)**

Baixa tensão DC **4 a 30 Vdc (corrente contínua)**

A configuração dos jumpers para cada uma destas configurações é dada abaixo, respeitando a convenção de 0 indicando jumper aberto e 1 indicando jumper fechado:

		Alta Tensão		Baixa Tensão	
Entrada	Jumpers	DC	AC	DC	AC
E1	JP2, JP6	00	01	10	11
E2	JP3, JP7	00	01	10	11
E3	JP4, JP8	00	01	10	11
E4	JP5, JP9	00	01	10	11

As Expansões de Entrada/Saída µDX110 são fornecidas de fábrica com as entradas programadas para alta tensão AC.

Parte

Instalações Industriais e Transitórios de Tensão

Em uma instalação industrial é comum a ocorrência de transitórios de alta tensão. Estes transitórios podem afetar o funcionamento de equipamentos eletrônicos, que normalmente operam em tensões baixas, como 3,3V ou 5 V. Os transitórios são originados da comutação de cargas indutivas, descargas atmosféricas e transitórios oriundos da rede elétrica, além de descargas eletrostáticas.

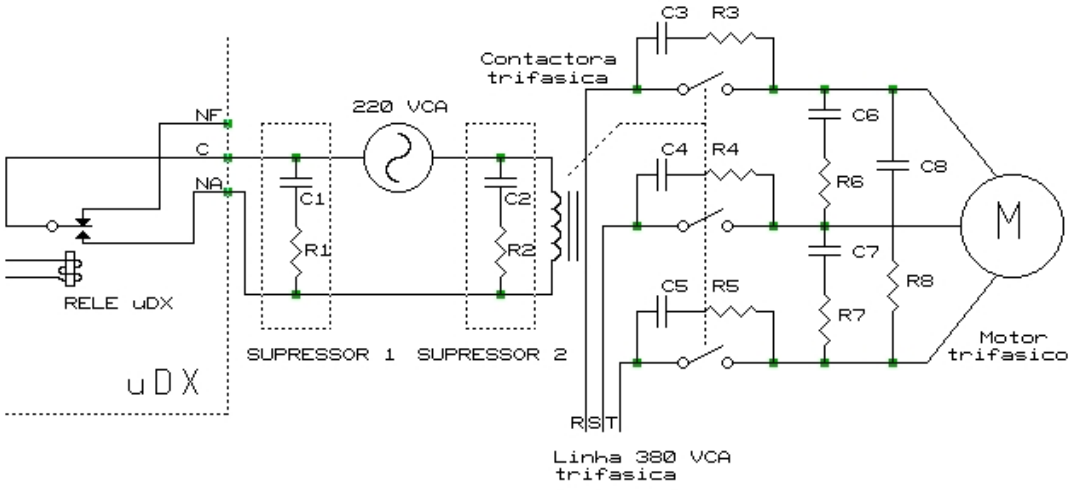
O controlador µDX100 possui vários componentes internos para protegê-lo de EMI (interferência eletromagnética), EMS (descargas eletrostáticas), e ainda descargas elétricas de alta energia. Ele foi testado com descargas de 1500V e energia de 3 joules em ciclos de 100 descargas, cada uma com duração total de cerca de 20µs (potência acima de 100KW). Isso garante um equipamento robusto e adequado a ambientes hostis, como comumente encontrados na indústria.

Cargas Indutivas

O controlador programável µDX Série 100 possui considerável proteção contra transitórios de alta tensão, como gerados na comutação de cargas indutivas (motores, solenóides, contactoras, etc.). Mas é aconselhável instalar supressores R-C (resistor + capacitor) para absorver e dissipar a força contraeletromotriz armazenada nas cargas indutivas, que é liberada na abertura dos contatos dos relés. Estes supressores, além de evitar problemas no funcionamento do controlador programável, aumentam muito a vida útil dos contatos de relés, pois evitam o arco elétrico criado quando esses contatos abrem.

O supressor R-C é formado por um resistor e um capacitor em série, ligados em paralelo com o contato do relé e em paralelo com a carga indutiva. O capacitor absorve a energia liberada pela carga indutiva ao repentinamente ter a corrente interrompida e o resistor serve para dissipar esta energia (amortecer o circuito L-C formado).

Teoricamente, bastaria um supressor R-C em paralelo com a carga que gera o transitório. Entretanto, normalmente os cabos de conexão à carga são longos. Ora, cabos longos são indutivos. Assim, é aconselhável também um filtro R-C junto ao relé do µDX100 para suprimir o transitório gerado pelos cabos de conexão, no caso desses serem longos (maiores que 5 metros).



Muitas cargas possuem uma corrente de manutenção (que as mantêm acionadas) muito baixa. Por exemplo, um solenóide para uma válvula pneumática, pode acioná-la somente aplicando uma tensão próxima de 220VCA. Mas, uma vez acionada a válvula, tensões muito mais baixas aplicadas ao solenóide a mantém neste estado. Ou seja, o supressor R-C instalado em paralelo com os contatos do relé do µDX100 deve ter uma corrente quiescente baixa, sob risco de manter a carga indefinidamente energizada. Uma solução é fazer C2>>C1, de forma que a corrente do supressor 2 seja muito maior que a do supressor 1, garantindo a desenergização da carga.

No caso de acionamento de cargas não-indutivas (lâmpadas, resistências, etc.) basta a instalação do supressor em paralelo com os contatos do relé do µDX100.

> ***ATENÇÃO:*** *Se o supressor estiver em uma instalação de corrente alternada existirá uma corrente quiescente devido ao capacitor. Isto pode acasionar choques elétricos ao manipular a carga, mesmo com os contatos do relé do µDX210 abertos.*

Se existirem contactoras acionando cargas indutivas (como motores, por exemplo), além dos supressores no contato do relé e em paralelo com a bobina da contactora, é interessante instalar supressores R-C em paralelo com a carga acionada pela contactora e em paralelo com os contatos dessa. Estes supressores em paralelo com o motor e em paralelo com os contatos da contactora evitam a formação de arco voltaico ao abrir a contactora, e portanto aumentam a vida útil dessa.

A seguir temos uma tabela com algumas cargas indutivas e os respectivos supressores R-C recomendados:

	Capacitor	**Resistor**	**Varistor**
Contactoras (bobina)	1µF/630V	100Ω,10W	-
Contactoras (contatos)	0,47µF/630V	100Ω, 5W	-
Relés µDX100 (contatos) (p/275V)	0,1µF/630V	100Ω,1/2W	S07K275
Solenóides (bobina) (válvulas pneumáticas, p/exemplo)	1µF/630V	100Ω,10W	-
Motores(bobina)	1µF/630V	100Ω,10W	-

Note que os capacitores devem ser adequados ao regime de corrente alternada com altas correntes de "ripple", como os capacitores de poliéster metalizado. Por exemplo, a série MKT da Siemens (MAC FITA - B32232).

A Dexter comercializa supressores de ruído com as especificações acima, sendo designados como:

Filtro pequeno	Capacitor 0,1µF + Resistor 100R 1/2W + Varistor 275V.
Filtro médio	Capacitor 0,47µF + Resistor 100R 5W.
Filtro grande	Capacitor 1µF + Resistor 100R 10W.

Gabinete

A caixa do controlador programável µDX100 e de seus acessórios não é adequada a uma instalação industrial, pois não tem proteção contra pó ou água, além das conexões elétricas ficarem expostas. Portanto, normalmente estes equipamentos são instalados dentro de outro gabinete metálico e aterrado (tanto o fundo quanto a tampa).

> ***ATENÇÃO:*** *O µDX100 e seus periféricos (Expansão de Entradas/Saídas, Modem, IHM, etc.) não devem ser instalados no mesmo painel das contactoras. Além disso, os cabos das entradas e das saídas do µDX100 e da Expansão devem ser mantidos o mais afastados possível. Cabos de alimentação elétrica para o equipamento e o cabo da rede DXNET também devem ser mantidos afastados dos cabos de saídas. Os cabos da rede DXNET, assim como os cabos das entradas do µDX100 e os de alimentação elétrica para os equipamentos não devem ser misturados a cabos de força.*

Utilize sempre duas calhas ou eletrodutos metálicos aterrados para agrupar os cabos de sinal. Onde for impossível evitar o cruzamento entre cabo de sinal e de força, este deve ser feito à 90°, minimizando o acoplamento eletomagnético entre ambos. Nunca coloque cabos de sinal e de força em paralelo. Neste caso use um eletroduto metálico aterrado para isolar os cabos de sinal (entradas do µDX, DXNET, etc.).
Ainda referente ao gabinete para acondicionar µDX+periféricos, convém envolver os cabos de força que chegam as saídas da Expansão com uma malha metálica, evitando irradiação eletromagnética dentro do gabinete. Outra providência interessante é utilizar um lado do gabinete para a fiação de entradas de baixa tensão, entradas e saídas analógicas, rede DXNET e sensores, e o outro lado (o mais afastado possível) para a cablagem de força (saídas de relé).

Parte

Programação em PDE - Utilização do PG

Acompanha o Controlador µDX100 um CD com o software de programação PG (Programador Gráfico), composto de Editor e Compilador, para computador IBM-PC compatível. Este software deve rodar sob plataforma Windows (Windows98, Windows Millenium, Windows NT, Windows 2000, Windows XP,Windows Vista ou Windows 7).

O PGEditor100 é uma ferramenta com a qual se pode elaborar ou modificar programas para o µDX100 (em linguagem PDE) e que, em conjunto com o programa Compilador PG (que acompanha o pacote de software) permite monitorar e interferir em qualquer µDX100 conectado na Rede Local DXNET.

Esta linguagem PDE - Programação por Diagrama Esquemático - foi desenvolvida pela DEXTER como um meio de programação intuitiva, de fácil compreensão e que dispensa conhecimentos especializados. O novo PG mantêm a filosofia de programação inaugurada pelo programa PG em sistema operacional DOS para linha de controladores µDX100, mas expande bastante suas possibilidades, ao permitir múltiplas janelas, conexão com transporte de valor de variáveis, e muito mais blocos.

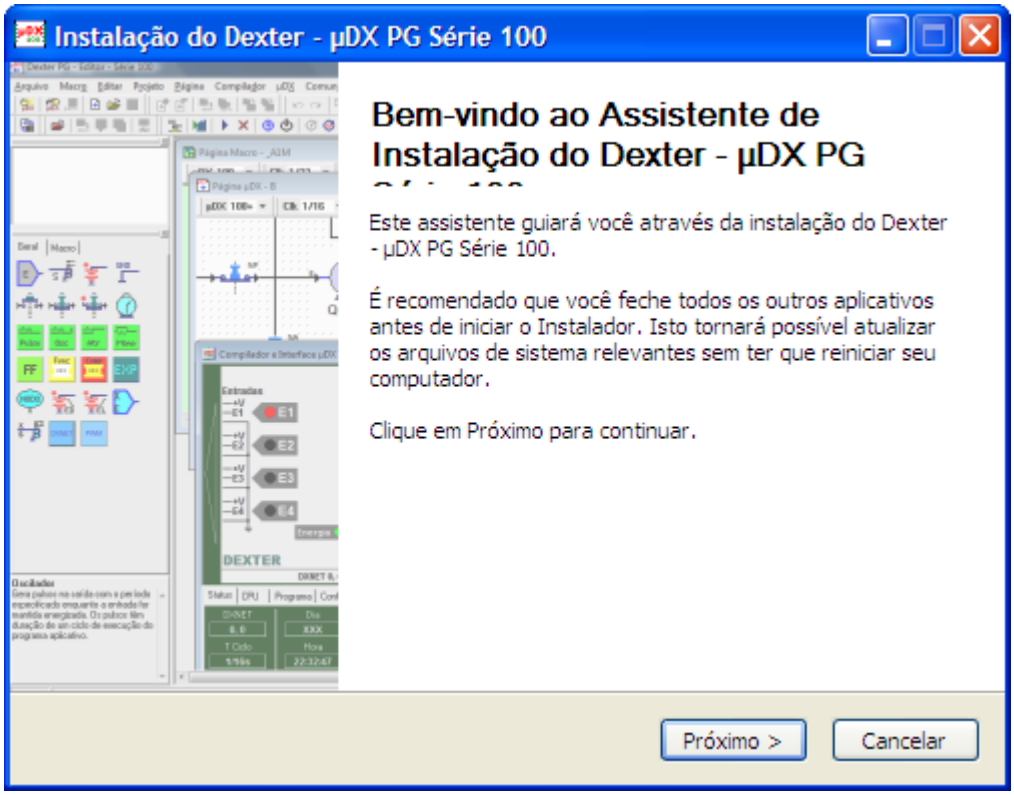
Para elaborar um programa basta "pegar" os blocos de instrução que sejam necessários (dispostos em forma de Menu), colocando-os depois na área de programação. Depois liga-se uns aos outros com fios e a representação artística final é a de um diagrama esquemático, com as chaves, relés, temporizadores, fios, fontes de energia e tudo o mais que o programa precisar. Textos podem ser inseridos no desenho para explicar alguma operação ou indicar a finalidade das entradas e saídas. E quando for necessário alterar o programa basta apontar o mouse e mover linhas e blocos que se desejar, apagar ou inserir novos blocos e linhas.

Finalmente, através de um único comando, o programa PGEditor100 pode chamar o módulo de Compilação, capaz de compilar e enviar o programa a qualquer µDX100 conectado a rede DXNET, sem perda de tempo.

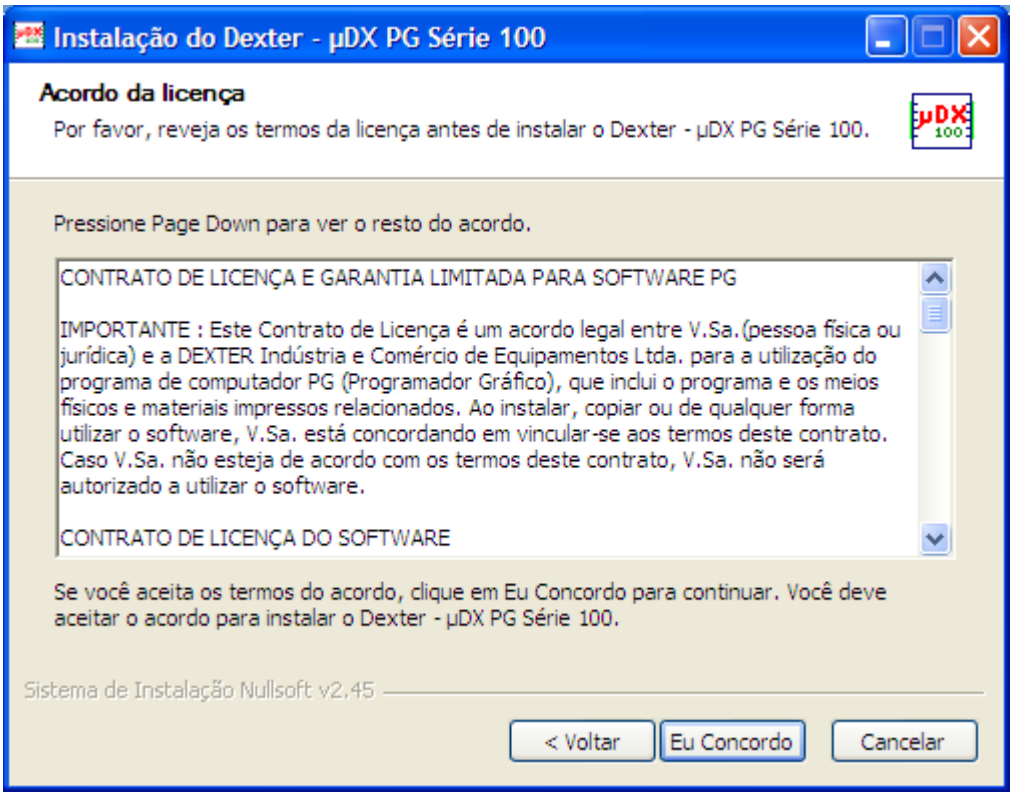
Além disso, o Compilador permite monitorar valores de variáveis, forçar outros valores ou ainda ligar algum nodo de qualquer µDX100. Por exemplo, através de um só comando pode-se ligar um relé de um µDX100.

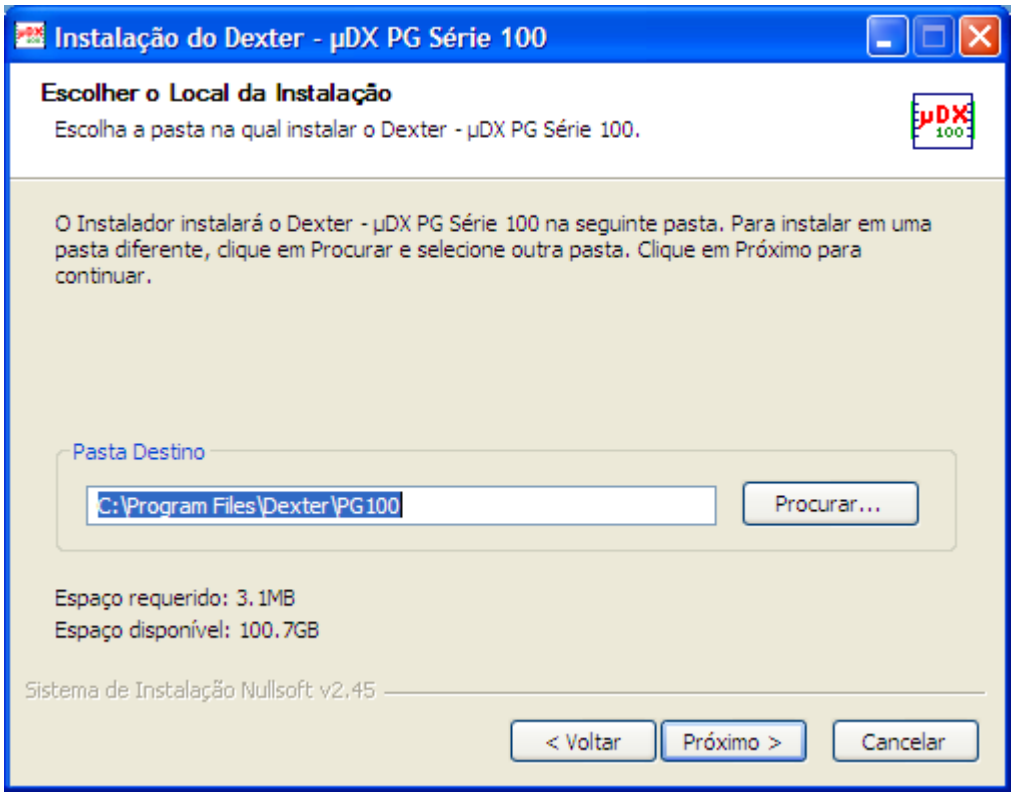
Instalação do software PG

Para instalar o software PG (Programador Gráfico) basta inserir o CD que acompanha o Controlador Programável µDX100 no leitor de CD do computador IBM-PC compatível. É recomendável o uso de computador com, no mínimo, 100MHz de clock. O PG roda sob sistemas operacionais Windows 98, Windows Millenium, Windows XP, Windows Vista e Windows 7. Caso a tela de inicialização não surja abra o diretório do CD via Windows Explorer e rode o programa **Instalar.exe** existente no CD.



A seguir, é apresentado um contrato de licença de uso para o software PG. Caso haja alguma cláusula do referido contrato que não seja aceitável de seu ponto de vista clique em [Cancelar] e entre em contato com a Dexter. Se o contrato for aceito o software PG será instalado, normalmente em c:\Arquivos de Programas\Dexter\PG100.





O instalador irá criar dois atalhos na tela do Windows, um para o Editor PG100 e outro para o Compilador PG100:

Atenção: Estes dois atalhos chamam o mesmo programa - PGEditor100.exe, apenas com diretivas diferentes, de forma a abrir na edição ou na compilação de programas aplicativos para o µDX100. Cuidado para não abrir duas vezes o PG inadvertidamente. Para passar do Editor para o Compilador e vice-versa use a tecla [Compilador/Fontes] (F5) existente no ambiente integrado. Caso seja chamado um atalho (por exemplo, µDX PG Editor Série 100) e depois o outro (µDX PG - Compilador Série 100) serão executados dois softwares PG simultaneamente no computador.

Compatibilidade com Versão DOS

O software PG em ambiente Windows não é compatível com o software PG para ambiente DOS. As modificações necessárias para utilizar toda flexibilidade disponível em ambiente Windows, como teclas de Desfazer (UNDO), múltiplas janelas de edição, etc., tornaram os programas aplicativos gerados em um ambiente incompatíveis com o outro. Para migrar um programa elaborado em PG em DOS para o novo ambiente é preciso redesenhá-lo no PG em Windows.

Atualização do PG

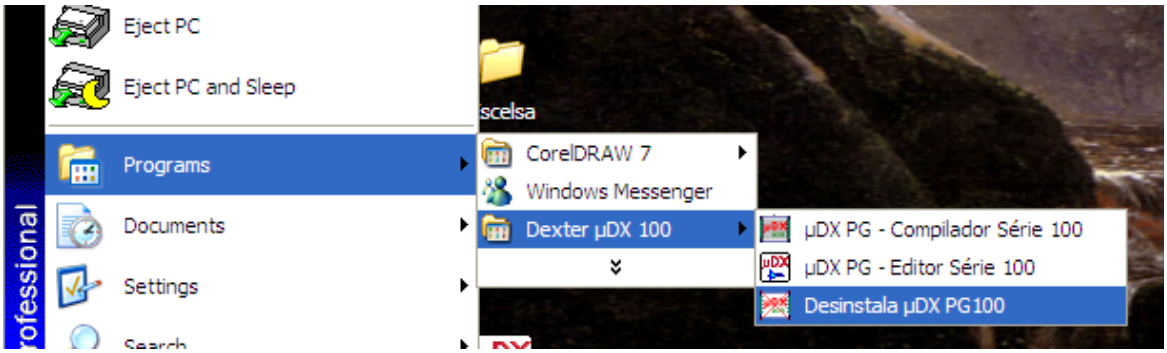
O software PG verifica no site da Dexter - www.dexter.ind.br - se existem versões mais recentes para sua atualização. Neste caso se recomenda a desinstalação da versão atual do PG e instalação da versão atual. Para isso siga os procedimentos a seguir:

Passo 1

Feche o software PG, salvando os arquivos porventura criados no mesmo.

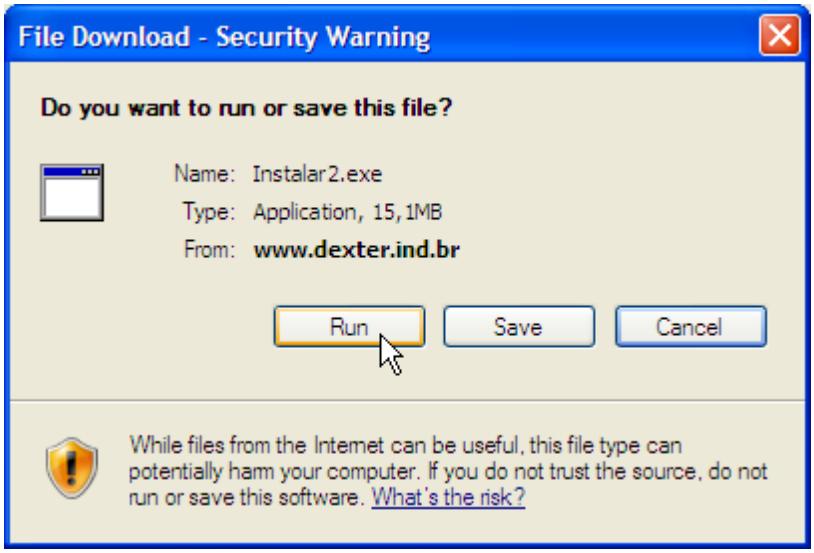
Passo 2

Utilize o ícone de desinstalação existente no menu **Iniciar** → **Programas** → **Dexter µDX** → **Desinstala µDX PG** do Windows:



Passo 3

Abra seu navegador de Internet (browser) e vá em http://www.dexter.ind.br/pg2.htm e faça o download da última versão do software PG. Pode-se optar por executar o programa:



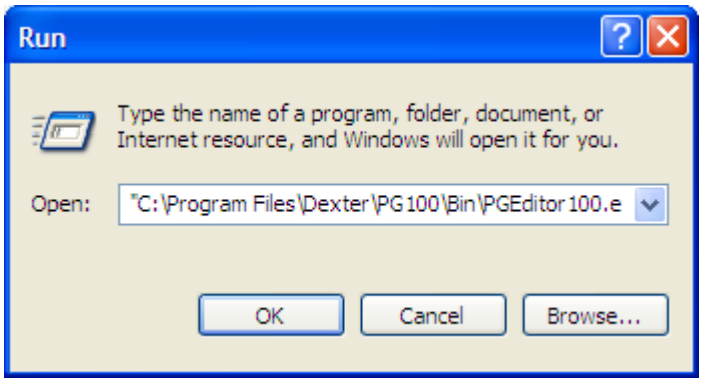
Passo 4

Siga os passos de instalação descritos no item Instalação do software PG. Está pronta sua atualização do PG! Note que todos os programas aplicativos para o PG que existiam nos diretórios de instalação (c:\Arquivos de Programas\Dexter) são preservados.

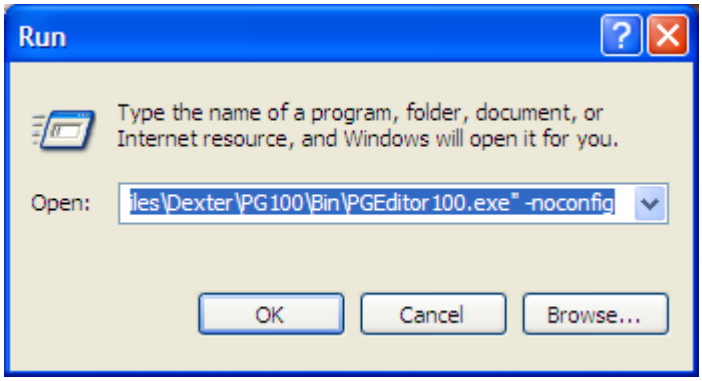
Diretivas de Linha de Comando

O programa **PG** (Programador Gráfico) permite uma série de diretivas a nível de linha de comando. Isso pode ser útil no caso de haver degradação em arquivos de configuração ou de bibliotecas, impedindo que o programa inicie.

Para rodar o programa via linha de comando vá em **Iniciar** → **Executar...** no Windows (**Start** → **Run**, caso seu sistema operacional esteja em inglês) e selecione o programa **PGEditor100.exe**:



Digite após as aspas finais um espaço em branco e a diretiva desejada. Por exemplo, para que o PG ignore as configurações no Registro do Windows, agindo como se estivesse rodando pela primeira vez desde a sua instalação:



Esta diretiva pode resolver problemas oriundos de erros na instalação do PG, ou instalação de múltiplas versões do PG, restaurando o registro do PG no sistema operacional Windows.

Parâmetros na linha de comando

-newpack Faz com que o editor ignore a ausência do arquivo de lista de bibliotecas.

-noconfig Apaga as configurações existentes no registry, se houver, e as ignora, agindo como se estivesse iniciando pela primeira vez.

-odxp %1 Abre o projeto %1.

-odxg %1 Abre o arquivo de página %1.

-odxm %1 Abre o arquivo de macro %1.

-oudp %1 Abre o arquivo compilado %1.

-comp Abre a janela do compilador.

Tipos de Arquivo

O software PG para µDX100 gera uma série de arquivos com sufixos distintos. São eles:

Arquivos sufixo:	Função do Arquivo:
D1P	**Arquivo de projeto do PG (várias páginas D1G reunidas em um único projeto).**
D1G	**Página de programação do PG para µDX100.**
U1P	**Página ou projeto pré-compilado pelo PG para µDX100.**
U1I	**Arquivo com configurações de hardware do µDX100 (mesmo nome do arquivo U1P associado).**
UDG	**Layout de monitoração para programa U1P**
D1M	**Página de Macro-célula do PG para µDX100.**
D1B	**Macro-célula pré-compilada pelo PG para µDX100.**
BMP	**Arquivo bit-map com imagem da página de programação, gerada pela opção Arquivo → Exportar...**
JPG	**Arquivo bit-map compactado com imagem da página de programação, gerada pela opção Arquivo → Exportar...**

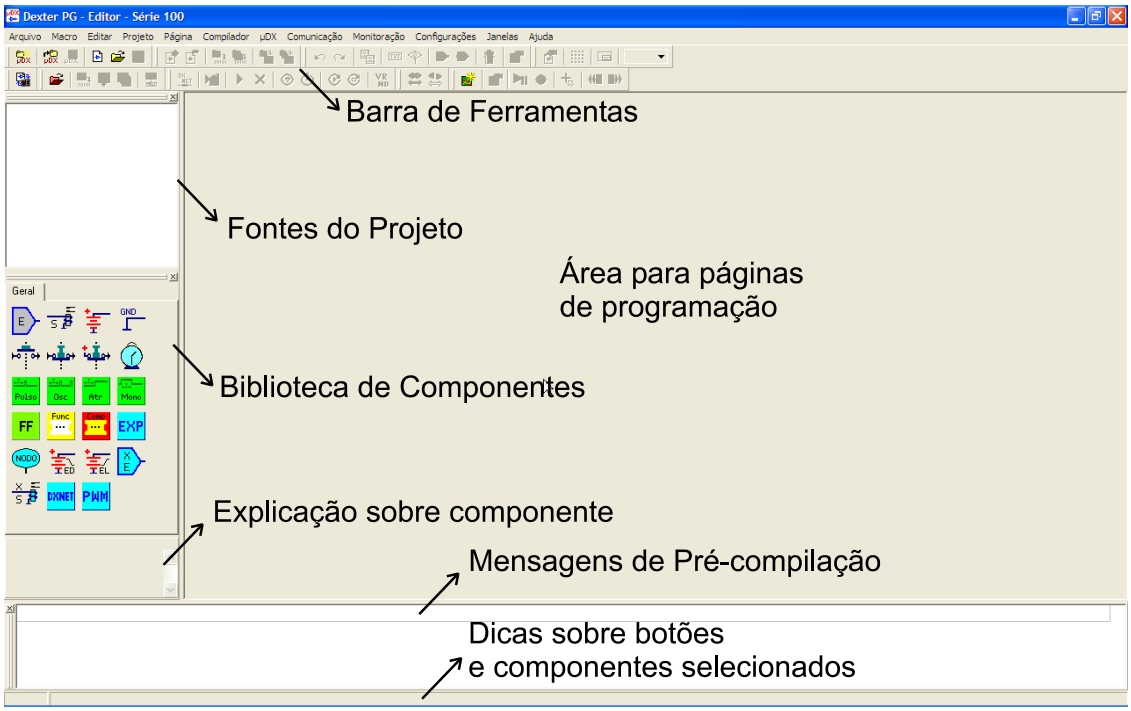
Teclas de Operação do Editor PG

Para utilizar o PG é necessário empregar um microcomputador compatível com IBM-PC com sistema operacional Windows 98 ou mais recente (Windows Millenium, NT, 2000, XP, Vista, 7).

Caso ainda não tenha instalado o software PG siga as instruções constantes em capítulo anterior deste manual. O CD que acompanha o µDX100 permite a instalação do software PGEditor100, e já gera atalho na tela principal do microcomputador para o programa.

Ao rodar o Editor PG a primeira tela que aparece mostra uma foto do µDX100 e algumas informações, como a versão do software. Após alguns segundos surge a tela do PG.

A tela principal do Editor PG apresenta uma grande área retangular que é onde se localizam as páginas com os programas, uma área onde aparecem os blocos utilizados para o desenho (Biblioteca de Componentes) e uma área reservada para mensagens de pré-compilação. Além disso, uma barra de ferramentas completa o conjunto, assim como teclas com menus pop-down para acesso as várias opções do software (muitas das quais estão disponíveis também na barra de ferramentas).



Os menus pop-down seguem a seguinte convenção:

Funções imediatas: são designadas apenas pelo nome, sem terminação (exemplo: Arquivo → Salvar).

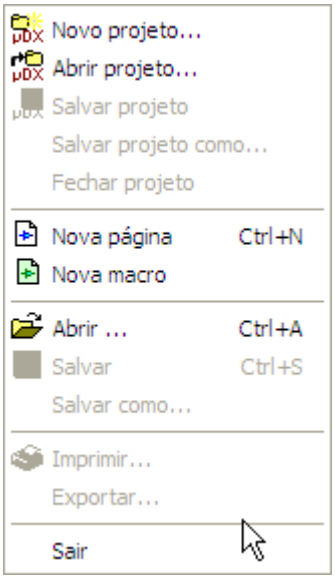
Funções que abrem janelas de opções: são terminadas por três pontos (exemplo: Arquivo → Salvar como...).

Funções que permitem selecionar um valor: são terminadas com seta ⇒ (exemplo: Página → Zoom).

Além disso, todas as funções que possuem teclas de atalho têm isto especificado após o nome da função (por exemplo, Página → Nova página Ctrl+N).

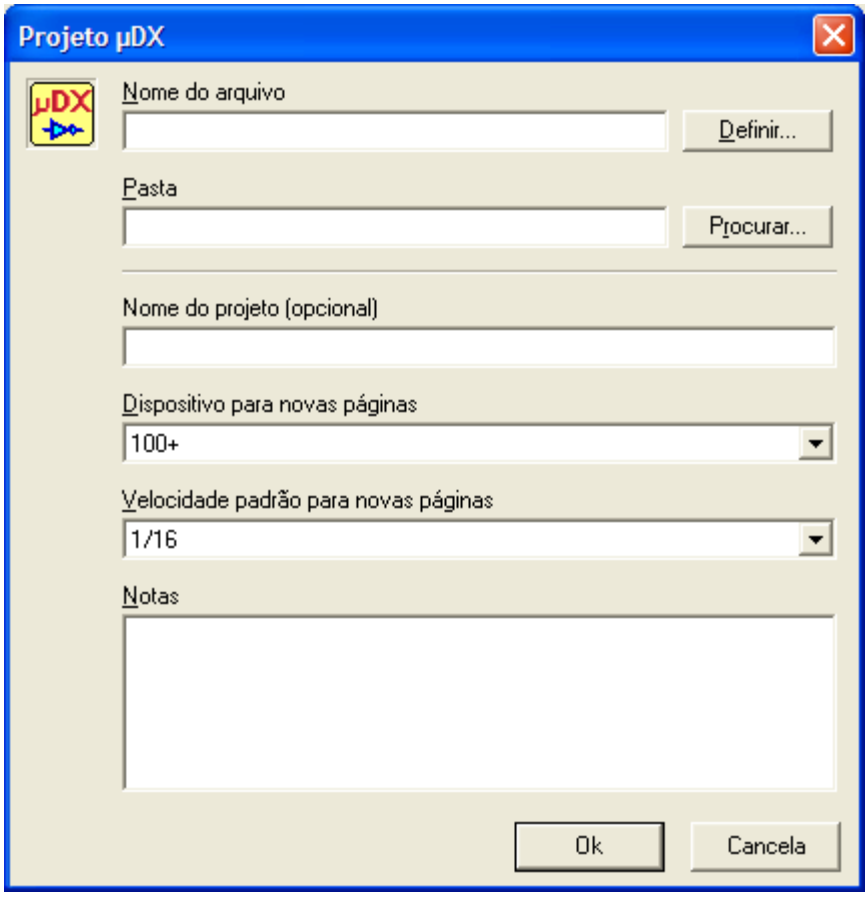
Menu Arquivo

Este menu permite gerenciar a leitura e salvamento de projetos ou páginas de programação, além de imprimir o programa e sair do Editor PG. Note que diversas funções estão disponíveis também na Barra de Ferramentas (as funções apresentam um ícone à esquerda que é usado também na barra de ferramentas), e três funções possuem teclas de atalho (Nova página - Ctrl+N, Abrir Página - Ctrl+A, e Salvar - Ctrl+S). As funções que não estão disponíveis aparecem em cinza.



Novo Projeto...

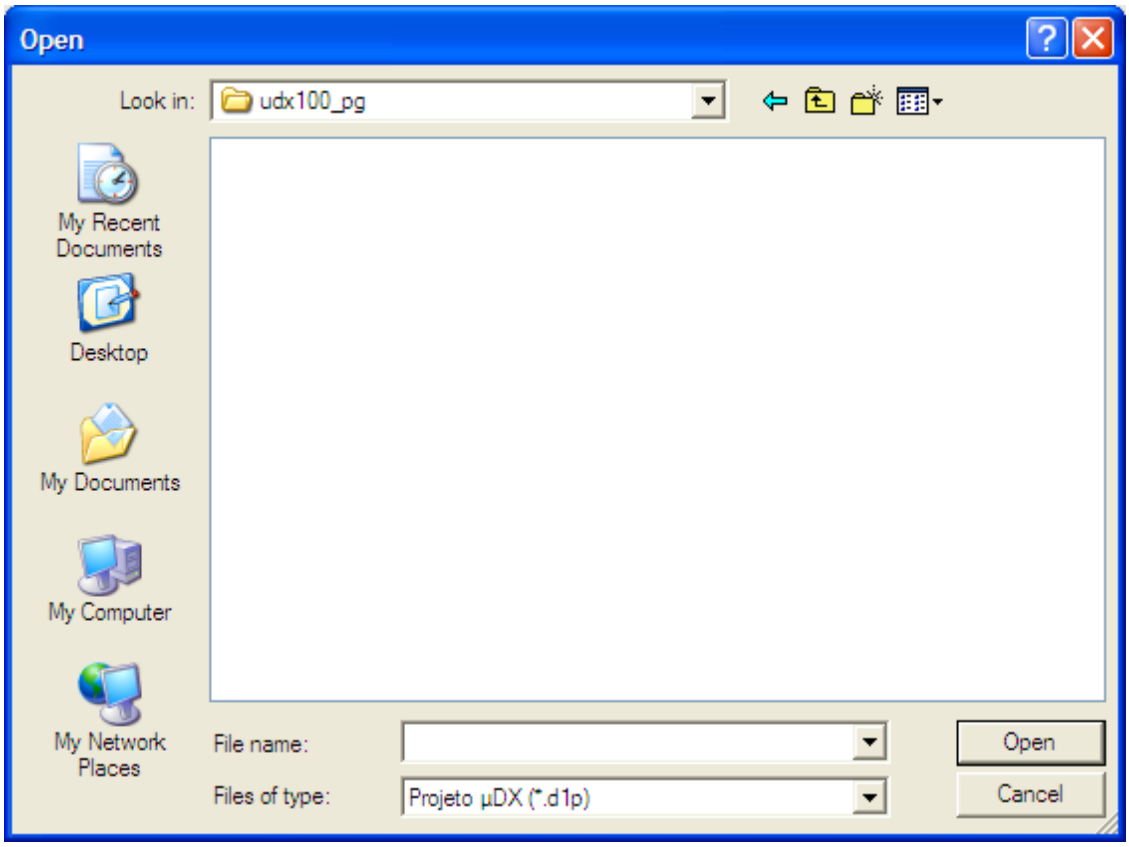
Esta tecla permite criar um novo projeto. Note que o Editor PG admite agrupar várias páginas de programação em um mesmo projeto. Com isso, ao se compilar o projeto todas as páginas serão compiladas e remetidas ao µDX100 como um programa único. Este recurso facilita sobremaneira a reutilização de partes de programas e a documentação de programas complexos. Ao selecionar [Novo Projeto...] surge a seguinte tela:



Esta tela permite determinar o nome do arquivo sufixo .d1p a ser criado, o diretório onde o mesmo será criado, um nome (opcional) para o projeto, o dispositivo alvo (µDX100 ou µDX100+), e ainda inserir alguns comentários sobre o projeto (notas).

Abrir Projeto...

Esta tecla abre um projeto já salvo anteriormente. Ao pressionar esta opção surge uma janela para seleção do projeto a ser lido:



Salvar Projeto

Esta tecla salva no disco rígido do microcomputador o projeto corrente. Isso inclui o projeto propriamente dito (arquivo sufixo .d1p) e todas as páginas que compõem o projeto (arquivos sufixo .d1g).

Salvar Projeto como...

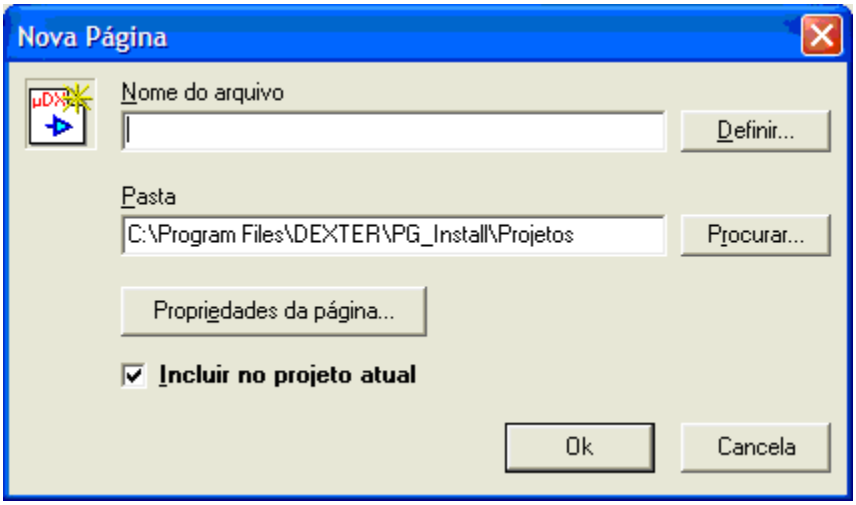
Esta tecla salva no disco rígido do microcomputador o projeto corrente, permitindo especificar um novo nome para o arquivo de projeto (arquivo sufixo .d1p). São salvas também todas as páginas que compõem o projeto (arquivos sufixo .d1g).

Fechar Projeto

Permite encerrar o projeto corrente. Caso o projeto tenha sofrido modificações e não tenha sido salvo o PG irá alertar o usuário, de forma a evitar perda de informações.

Nova Página (Ctrl+N)

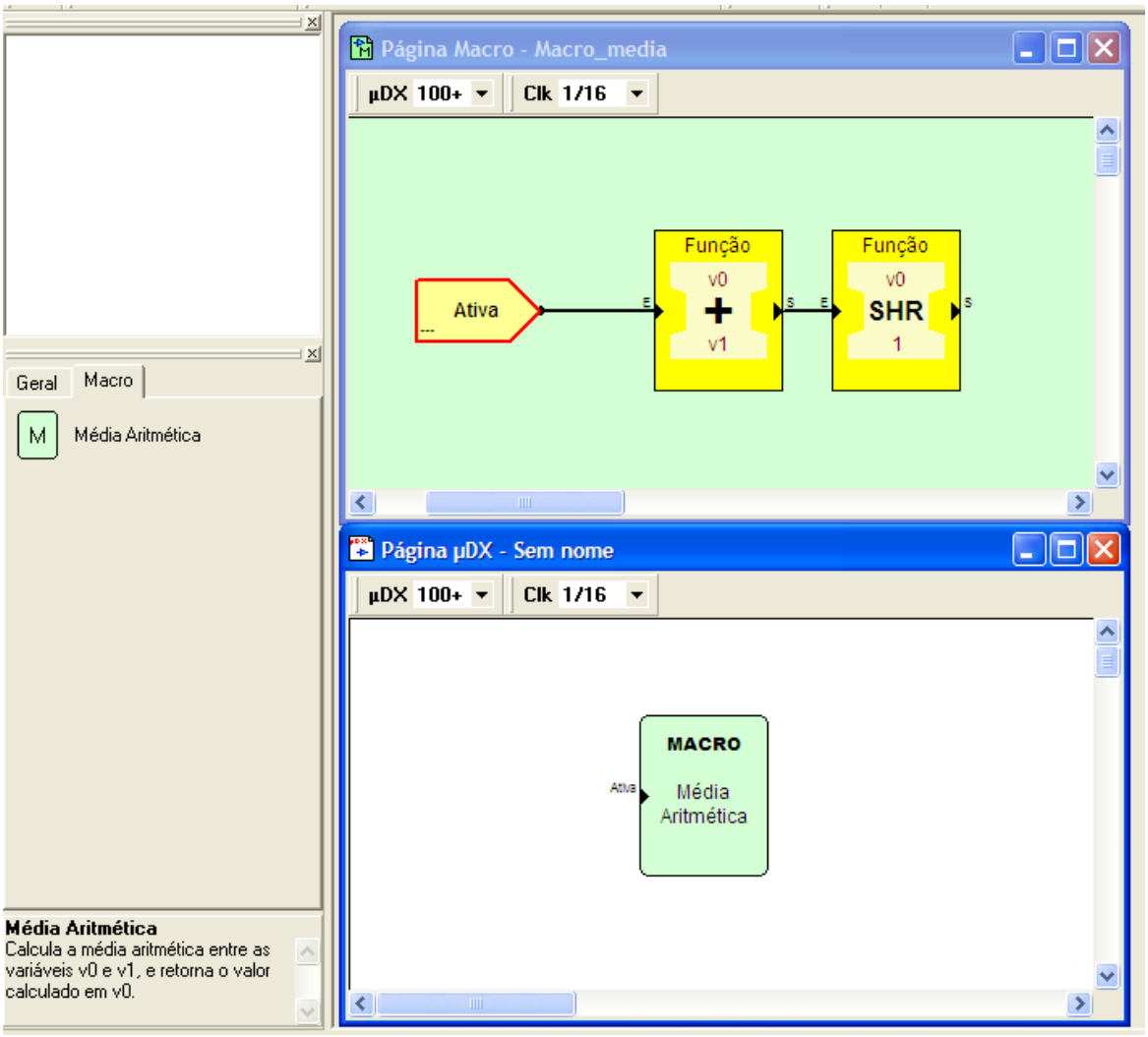
Abre uma nova página de programação na área de programação. Note que se houver um projeto aberto surgirá a tela abaixo:



Esta tela permite escolher um nome para a página, editar propriedades da página, e incluir a página no projeto corrente. Caso não exista projeto aberto a página é criada imediatamente, sem que surja a janela acima.

Nova Macro

Abre uma nova página de Macro na área de programação. Macros são programas aplicativos que serão compilados como um bloco único, permitindo sua utilização em outros programas. Note que a Macro se torna uma "caixa-preta", ou seja, nenhuma das variáveis e nodos internos estão disponíveis externamente, exceto se ligados às conexões de Macro (Entrada de Macro e Saída de Macro). A tela de programação de Macro (Página Macro) possui fundo verde, para diferenciar de tela de programação do µDX (Página µDX), que apresenta fundo branco. Note que é possível usar uma Macro dentro da definição de outra Macro. Com isso, é possível hierarquizar e organizar um programa, com Macros de crescente complexidade. Obviamente a Macro, apesar de visualmente se apresentar como apenas um bloco, ocupa na memória do controlador programável o número de blocos usados para sua criação. Abaixo um exemplo de Macro que calcula a média aritmética de duas variáveis. Veja que a Macro foi gerada na tela verde, e logo após ser compilada (Macro → Compilar e Salvar Macro) ela surje na Biblioteca de Componentes. Abaixo da janela de programação de Macro temos uma janela de programação µDX em que colocamos um bloco de Macro para cálculo de média. Ou seja, criamos um bloco para que o µDX100 calcule a média aritmética.

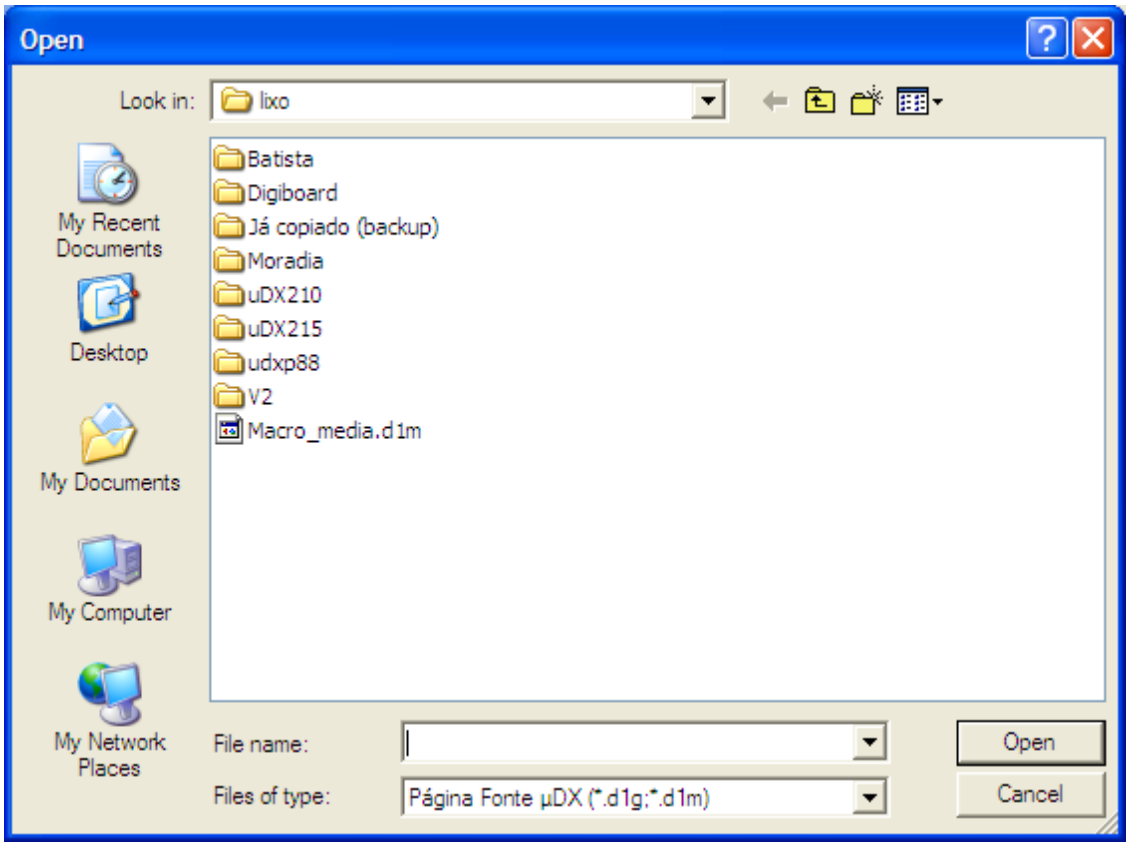


Exemplo de Macro: **Macro_media.d1m** (fonte); Macro_media**.d1b** (compilada).

Para salvar o programa que gerou uma Macro é necessário selecionar Salvar ou Salvar como... (veja as opções a seguir). Uma última observação: a função Nova Macro fica indisponível caso haja um projeto aberto no Editor PG. Para disponibilizar a função é necessário fechar o projeto.

Abrir... (Ctrl+A)

Abre uma página de programação previamente salva na área de programação. Note que, neste caso, a página aberta não é incluída no projeto (no caso de existir um projeto aberto). A janela de seleção da página tem o seguinte aspecto:



É possível também carregar uma página de programação de Macro. Para isso basta selecionar um arquivo fonte de Macro (sufixo .d1m).

Salvar (Ctrl+S)

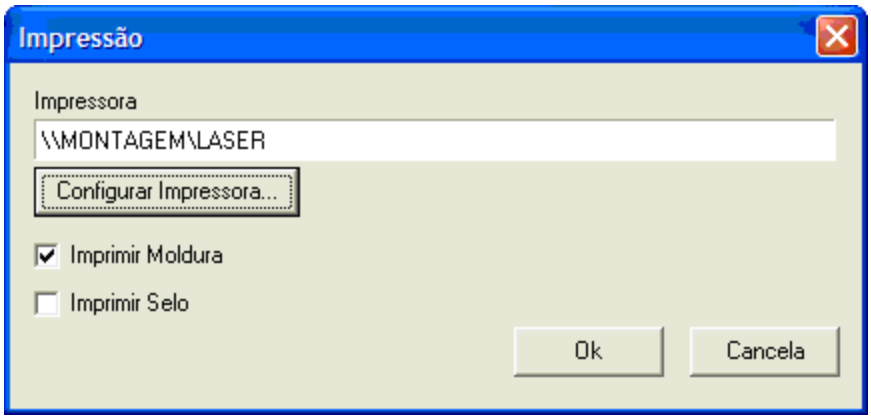
Salva no disco rígido do microcomputador a página µDX corrente (arquivo sufixo .d1g). Também permite salvar a página de Macro corrente (arquivo sufixo .d1m).

Salvar como...

Esta tecla salva no disco rígido do microcomputador a página µDX corrente, permitindo especificar um nome para o arquivo (arquivo sufixo .d1g). Também permite salvar a página de Macro corrente (arquivo sufixo .d1m).

Imprimir...

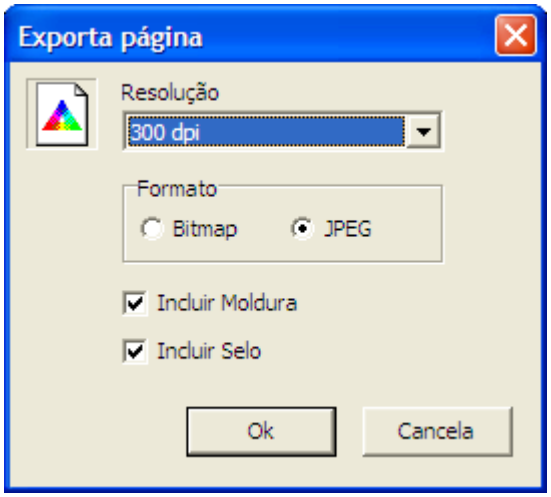
Imprime a página de programação corrente. Ao selecionar esta opção surge a tela:



É possível escolher a impressora (caso exista mais de uma impressora configurada no computador), configurá-la, e selecionar se devem ser impressas a moldura da página e o selo (quadro com informações no canto inferior direito da página).

Exportar...

Muitas vezes é necessário anexar a documentação de determinado projeto detalhes da programação implementada no controlador programável. Esta opção permite gerar um arquivo bit-map, compactado (JPG) ou não (BMP), com diversas resoluções. Normalmente uma resolução de 300 dpi (dots per inch) é suficiente, mas ela pode ser escolhida desde 150 dpi até 600 dpi. Como os arquivos JPG são cerca de 10 a 15 vezes menores que seus equivalentes em BMP aconselhamos seu uso sempre que possível:



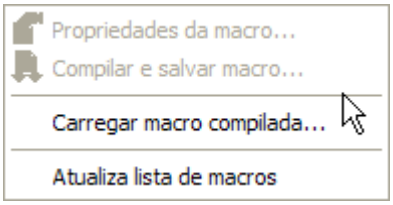
Note que para páginas de programação grandes a geração do arquivo BMP ou JPG pode levar vários segundos. Durante este período a janela acima ficará "congelada" no PG. Isso é normal. Aguarde a finalização do processo.

Sair

Permite encerrar o programa Editor PG.

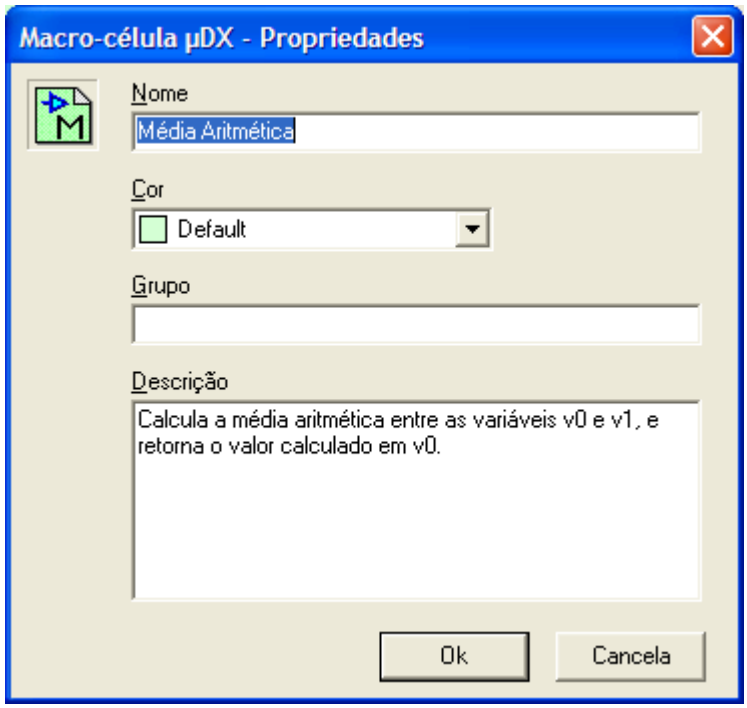
Menu Macro

Este menu permite editar propriedades de uma Macro, e também compilá-la, gerando um bloco utilizável nos programas aplicativos do µDX200. Além disso, é possível carregar macros compiladas por terceiros (arquivos sufixo .dmb) e atualizar a lista de Macros na Biblioteca de Componentes. As funções que não estão disponíveis aparecem em cinza.



Propriedades da Macro...

Esta tecla edita as propriedades de uma Macro. Ao selecionar esta função surge a seguinte tela (é necessário estar com uma página de Macro selecionada):



O campo **Nome** designa qual o nome da Macro. Este nome é que irá aparecer na Biblioteca de Componentes, assim como na própria representação como um bloco da Macro. Em **Cor** é possível especificar uma cor de fundo para a representação da Macro. Com isso, pode-se eleger determinadas cores para funções específicas (como é feito com os blocos do µDX100), facilitando a identificação das Macros. O campo **Grupo** permite agrupar as Macros na lista de Macros conforme sua funcionalidade. Caso seja mantido em branco as Macros não são agrupadas. Em **Descrição** pode-se incluir uma descrição sucinta da função da Macro.

Compilar e Salvar Macro...

Esta função compila a Macro, gerando um arquivo com sufixo .d1b. Note que o arquivo fonte de Macro possui sufixo .d1m. Ao compilar a Macro ela já é incluída na Biblioteca de Componentes (aba Macro). Note que esta opção é inibida no caso de existir um projeto aberto no PG. É necessário fechar o projeto antes de compilar Macro.

Carregar Macro Compilada...

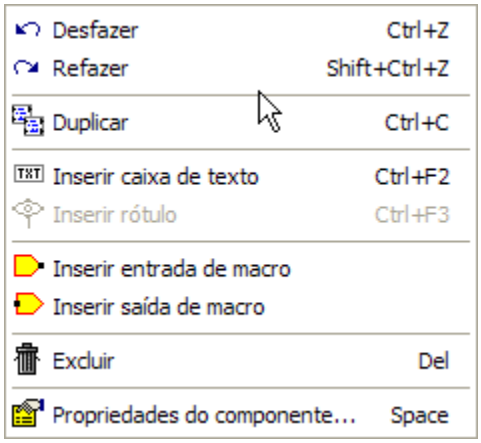
Pode-se carregar um arquivo .d1b de Macro compilada para a Biblioteca de Componentes do Editor PG. Com isso, é possível utilizar macros geradas por terceiros.

Atualiza Lista de Macros

Permite atualizar a lista de Macros existente na Biblioteca de Componentes do Editor PG.

Menu Editar

Este menu permite desfazer ou refazer operações, duplicar blocos ou áreas de programa, inserir caixas de texto ou rótulos, inserir entradas e saídas de Macros, excluir blocos ou áreas de programa, ou ainda visualizar propriedades de um componente selecionado. Note que todas funções estão disponíveis também na Barra de Ferramentas (com o mesmo ícone apresentado à esquerda), e também possuem teclas de atalho (representadas à direita). As funções que não estão disponíveis aparecem em cinza.



Desfazer (Ctrl+Z)

Desfaz a última operação efetuada no Editor PG. A tecla de Desfazer possui uma profundidade de cinco operações, ou seja, pode-se desfazer até as cinco últimas operações.

Refazer (Shift+Ctrl+Z)

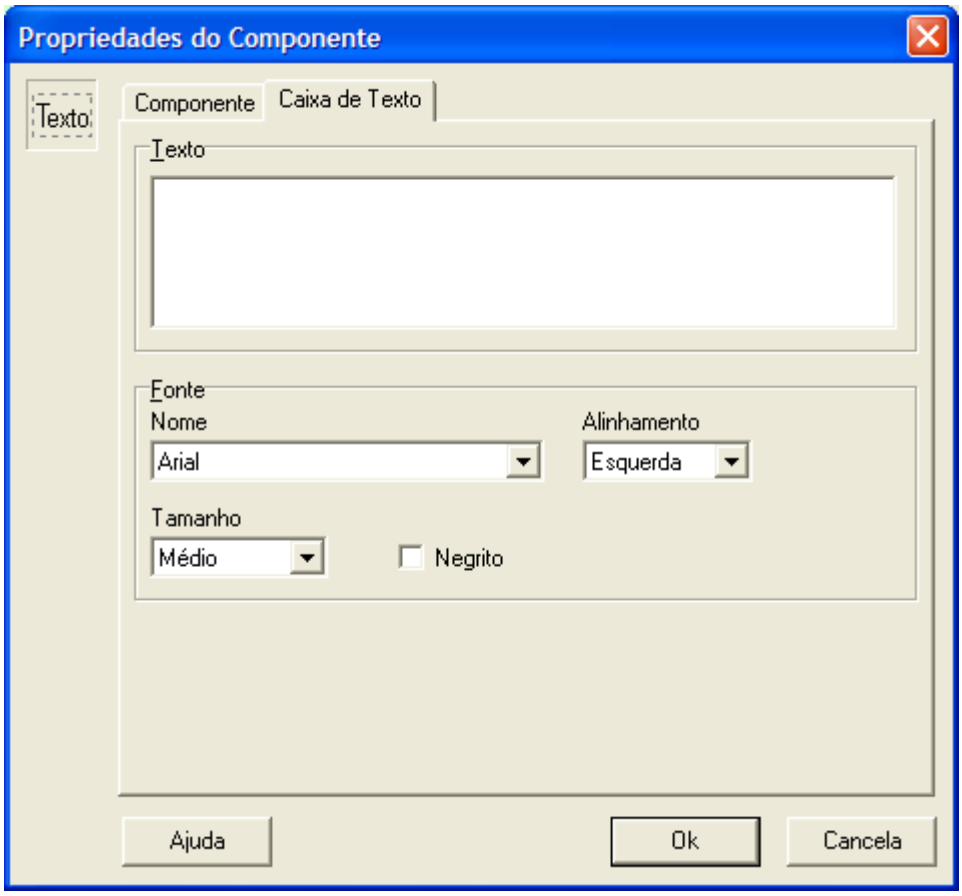
Refaz a última operação efetuada no Editor PG. A tecla de Refazer possui uma profundidade de cinco operações, ou seja, pode-se refazer até as cinco últimas operações

Duplicar (Ctrl+C)

Esta tecla permite duplicar um componente ou uma área selecionada do programa. Para selecionar um componente basta clicar sobre o componente com a tecla esquerda do mouse. Já para selecionar uma área mantenha pressionada a tecla esquerda do mouse e selecione um retângulo de seleção movimentando o mouse. Ao pressionar Duplicar os componentes presentes na área selecionada são duplicados, e podem ser incluídos tanto na própria página de programação que os originou quanto em outra página.

Inserir Caixa de Texto (Ctrl+F2)

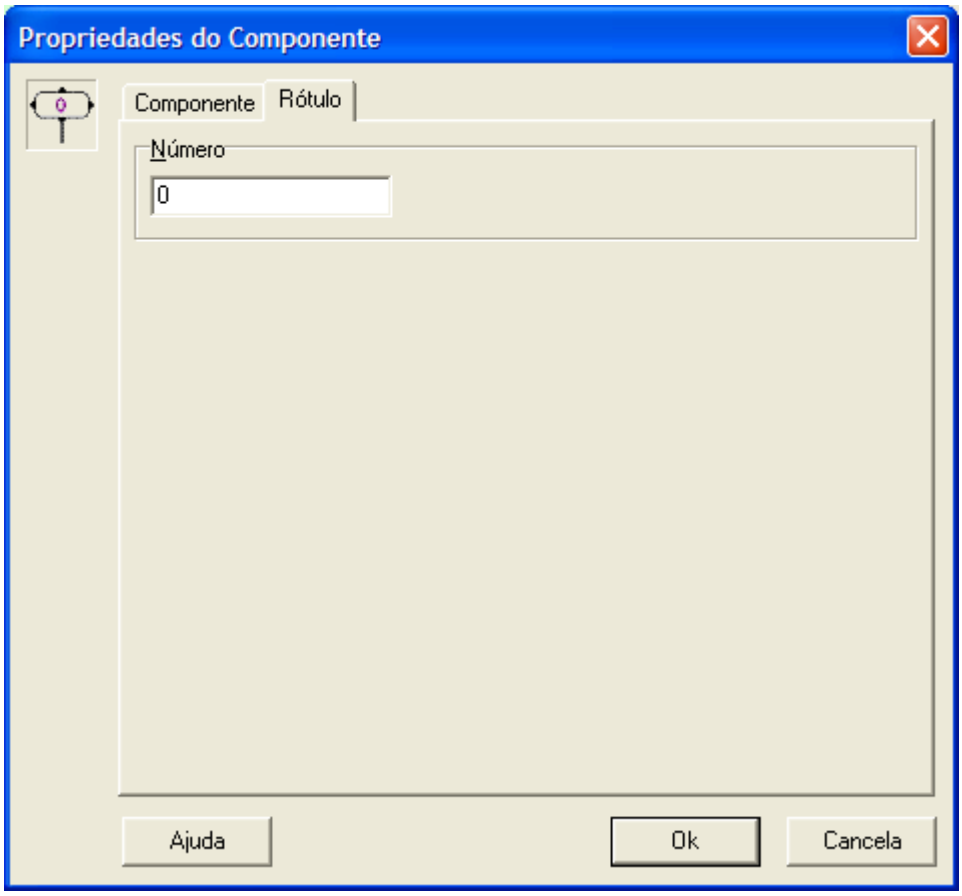
Insere uma caixa de texto no programa. Esta caixa de texto não é utilizada pelo programa, servindo exclusivamente como comentário para o programa aplicativo do µDX100. Para "largar" a caixa de texto sobre o programa basta clicar com a tecla esquerda do mouse sobre o local desejado. Para editar a caixa de texto, permitindo inserir o texto, clique com a tecla direita do mouse apontando para a caixa de texto. Deve surgir uma janela que permite, além da digitação do texto, a escolha da fonte de caracteres a ser usada, o tamanho dos caracteres, e se em negrito ou não, além do alinhamento do texto.



Para ajustar o tamanho da caixa de texto (de forma que o texto digitado caiba na caixa) basta selecionar a caixa e arrastar os cantos dessa, modificando suas dimensões.

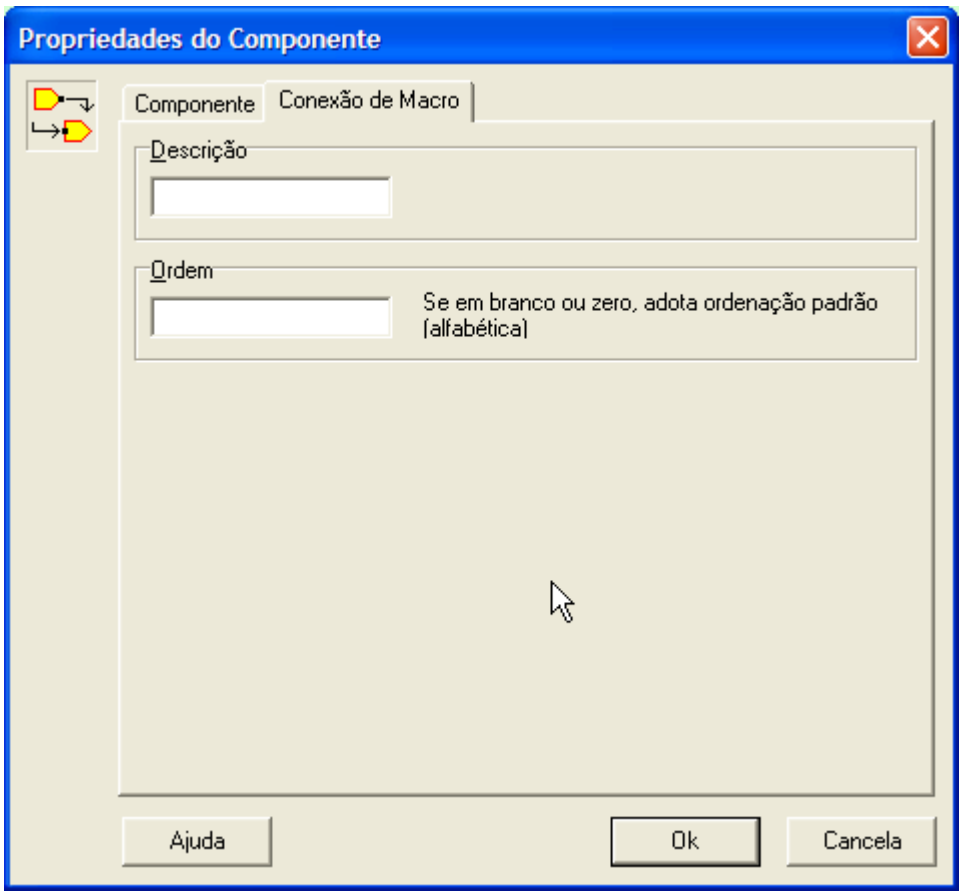
Inserir Rótulo (Ctrl+F3)

Insere um rótulo no programa. A função do rótulo é conectar diferentes pontos do programa entre si. O rótulo apenas conecta entre si todos os pontos com o mesmo número de rótulo (de 0 a 999), como se estivessem conectados por "fios", não atribuindo nenhum valor determinado ou nome para esta conexão. Ao editar o Rótulo surge a janela que permite atribuir um valor numérico ao mesmo. Todos os rótulos com mesmo valor numérico estarão conectados (mesmo em janelas de programação distintas de um mesmo projeto). Note que rótulo não é permitido em Macros. Desta forma, esta função estará inibida quando estiver selecionada uma página de Macro.



Inserir Entrada de Macro

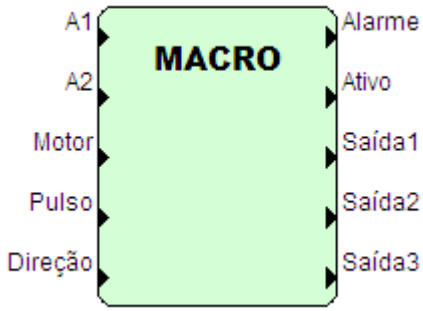
Insere um ponto de entrada para a Macro. Esta função somente fica ativa quando está selecionada uma página de Macro, já que não é possível usar **Entrada de Macro** em programas aplicativos do µDX100. Ao editar uma **Entrada de Macro** ou uma **Saída de Macro** surge a janela abaixo, que permite especificar um nome para a conexão da Macro, e escolher uma ordem para a colocação das conexões da Macro.



A **Ordem** em uma Entrada de Macro (ou Saída de Macro) especifica qual a ordem em que as conexões de entrada (ou saída) aparecerão no bloco de Macro. As entradas sempre são colocadas à esquerda do bloco de Macro, e as saídas à direita. Com isso, entradas e saídas são ordenadas independentemente. Caso o campo Ordem seja mantido em branco a ordenação será em ordem alfabética crescente. Caso seja colocado números no campo Ordem, os itens serão ordenados em ordem crescente destes números. É aceitável qualquer número inteiro (portanto, é possível usar-se números negativos). Se duas os mais entradas de Macro possuírem o mesmo número de ordem, elas serão ordenadas alfabeticamente. Por exemplo, digamos que foi gerada uma Macro com as seguintes conexões de entrada e saída:

Entradas de Macro		Saídas de Macro	
Descrição	**Ordem**	**Descrição**	**Ordem**
A1	1	Saída1	3
A2	1	Saída2	3
Motor	2	Saída3	3
Pulso	3	Alarme	1
Direção	4	Ativo	2

Neste caso o PG irá gerar um bloco de Macro como mostrado a seguir:



Inserir Saída de Macro

Insere um ponto de saída para a Macro. Esta função somente fica ativa quando está selecionada uma página de Macro, já que não é possível usar **Saída de Macro** em programas aplicativos do µDX100.

Excluir (Del)

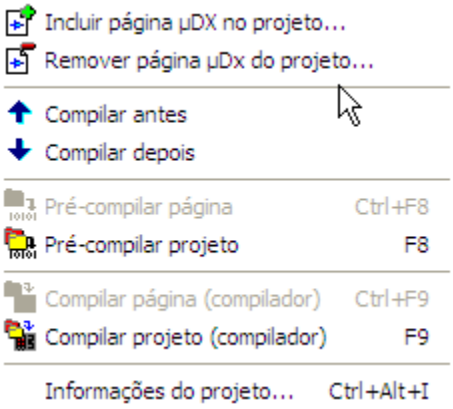
Exclui o componente ou a área selecionada do programa. Para selecionar um componente basta clicar sobre o componente com a tecla esquerda do mouse. Já para selecionar uma área mantenha pressionada a tecla esquerda do mouse e selecione um retângulo de seleção movimentando o mouse.

Propriedades do Componente (Space)

Apresenta as propriedades do componente selecionado. Note que as propriedades do componente também podem ser acessadas clicando sobre o mesmo com a tecla direita do mouse, exceto no caso das conexões ("fios"). A tecla direita, neste caso, gera uma conexão derivada da conexão existente. Já a tecla de Space (barra de espaço do teclado) sempre acessa as propriedades do componente.

Menu Projeto

Este menu permite incluir ou remover páginas do projeto, compilar página de programa ou o projeto inteiro (note que o Editor PG, na verdade, efetua uma pré-compilação), compilar o programa pré-compilado e abrir a janela do Compilador, ou ainda visualizar informações sobre o projeto. Note que todas funções estão disponíveis também na Barra de Ferramentas, à exceção de Informações do projeto (com o mesmo ícone apresentado à esquerda), e também muitas delas possuem teclas de atalho (representadas à direita). As funções que não estão disponíveis aparecem em cinza. Também é possível modificar a ordem das páginas em um projeto (compilar antes e compilar depois). Com este recurso é possível determinar quais blocos serão executados antes pelo µDX100.



Incluir página µDX no projeto...

Este item possibilita inserir páginas de programação µDX ao projeto corrente.

Remover página µDX no projeto...

Este item possibilita remover páginas de programação µDX ao projeto corrente.

Compilar antes

Permite modificar a ordem das páginas em um projeto. Basta selecionar a página e clicar em **Compilar antes** para fazê-la subir na lista de páginas do projeto. Note que blocos de inicialização de variáveis devem ser executados antes dos demais blocos do programa aplicativo. Então, a página do projeto que contém estas inicializações deve ser a primeira página do projeto.

Compilar depois

Permite modificar a ordem das páginas em um projeto. Basta selecionar a página e clicar em **Compilar depois** para fazê-la descer na lista de páginas do projeto.

Pré-compilar página (Ctrl+F8)

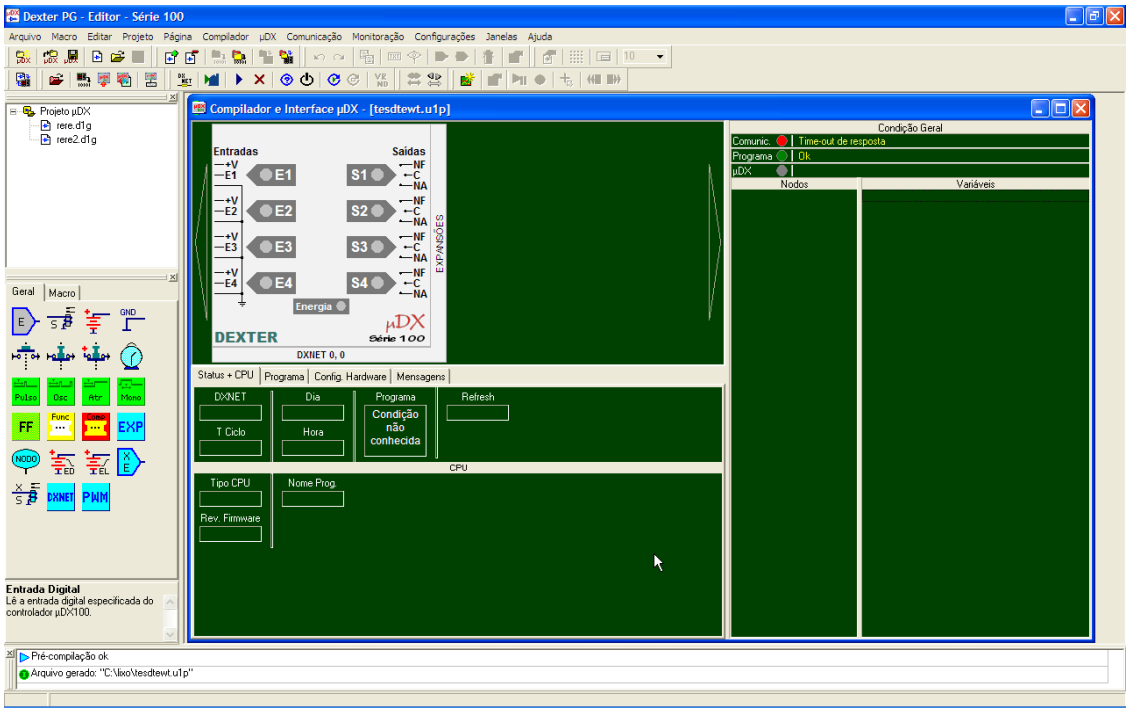
Efetua a pré-compilação apenas da página selecionada, gerando um arquivo de pré-compilação (arquivo sufixo .u1p).

Pré-compilar projeto (F8)

Pré-compila todas as páginas do projeto, gerando um arquivo de pré-compilação (arquivo sufixo .u1p) que inclui todas as páginas de programação µDX100 inclusas no projeto. As páginas podem compartilhar dados (basta que referências à mesma variável sejam feitas em diferentes páginas) ou conexões (usando rótulos com mesmo número em diferentes páginas).

Compilar Página (Compilador) (Ctrl+F9)

O Editor PG gera uma pré-compilação (arquivo sufixo .u1p). Este arquivo deve ser lido no Compilador PG (janela que inclui monitoramento do µDX100 e ferramentas para sua programação) para que seja efetuada a compilação final do programa aplicativo (que será transmitida para o controlador). Para isso é necessário abrir a janela do Compilador, e isso pode ser feito a partir do Editor PG via esta opção. Ao clicar nesta função é feita a pré-compilação da página de programação µDX (arquivo sufixo .d1g gera arquivo sufixo .u1p), a compilação propriamente dita, e é aberta a janela do Compilador e Interface µDX. Note que no cabeçalho desta janela aparece o nome do arquivo pré-compilado carregado:

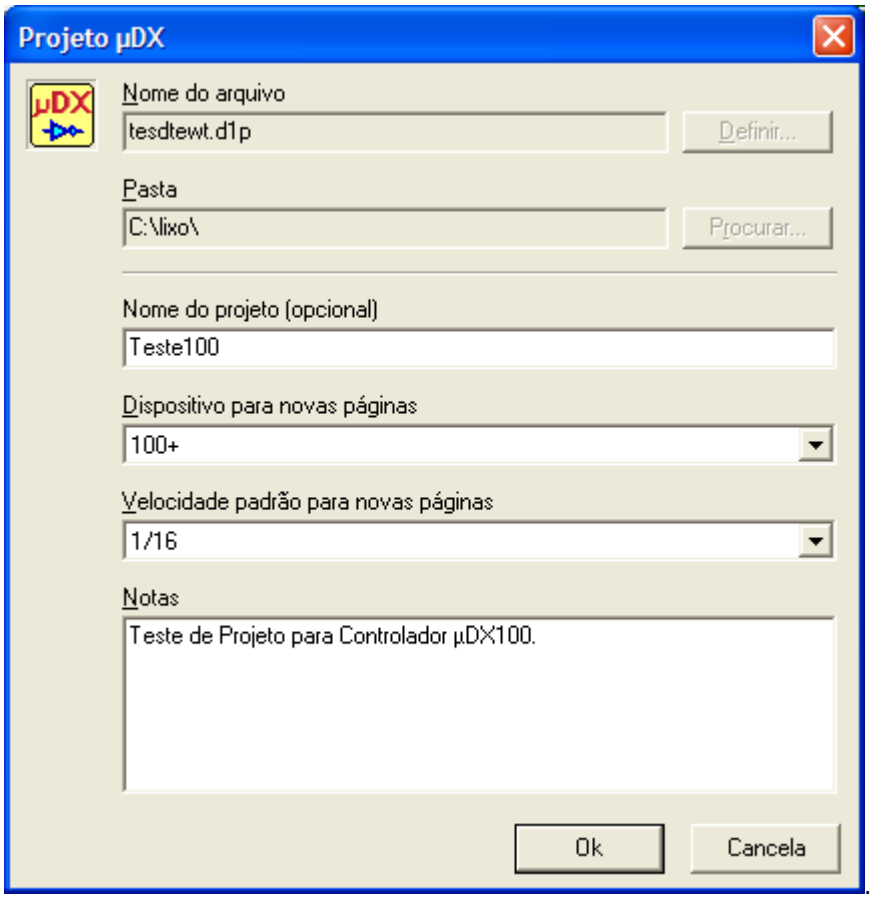


Compilar Projeto (Compilador) (F9)

O Editor PG gera uma pré-compilação (arquivo sufixo .u1p). Este arquivo deve ser lido no Compilador PG (janela que inclui monitoramento do µDX100 e ferramentas para sua programação) para que seja efetuada a compilação final do programa aplicativo (que será transmitida para o controlador). Para isso é necessário abrir a janela do Compilador, e isso pode ser feito a partir do Editor PG via esta opção. Ao clicar nesta função é feita a pré-compilação do projeto de programação µDX (arquivo sufixo .d1p gera arquivo sufixo .u1p), a compilação propriamente dita, e é aberta a janela do Compilador e Interface µDX. Note que no cabeçalho desta janela aparece o nome do arquivo pré-compilado carregado, como mostrado na figura anterior.

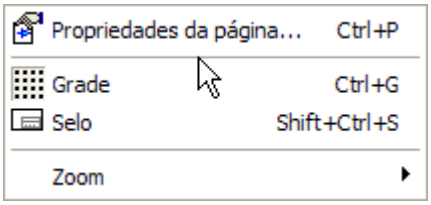
Informações do projeto... (Ctrl+Alt+I)

Retorna uma janela com informações sobre o projeto aberto no Editor PG, e permite editar estas informações.



Menu Página

Este menu permite editar propriedades da página, acionar ou inibir a grade na página, ativar ou desativar o selo da página (quadro no canto inferior direito com informações sobre a página de programação), e também selecionar o zoom de visualização. Note que todas funções estão disponíveis também na Barra de Ferramentas (com o mesmo ícone apresentado à esquerda), e também possuem teclas de atalho (representadas à direita). As funções que não estão disponíveis aparecem em cinza.



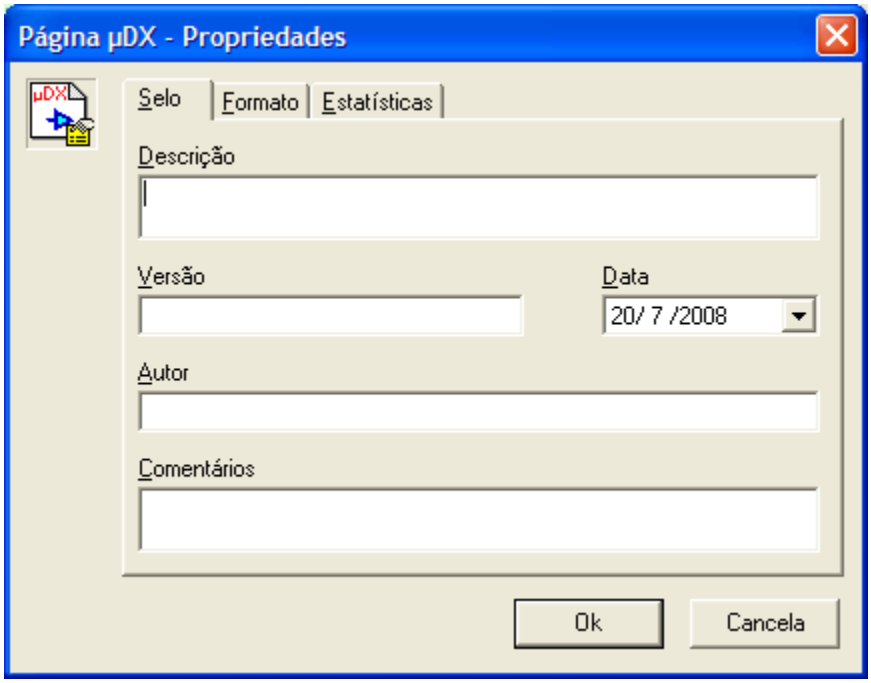
Propriedades da página... (Ctrl+P)

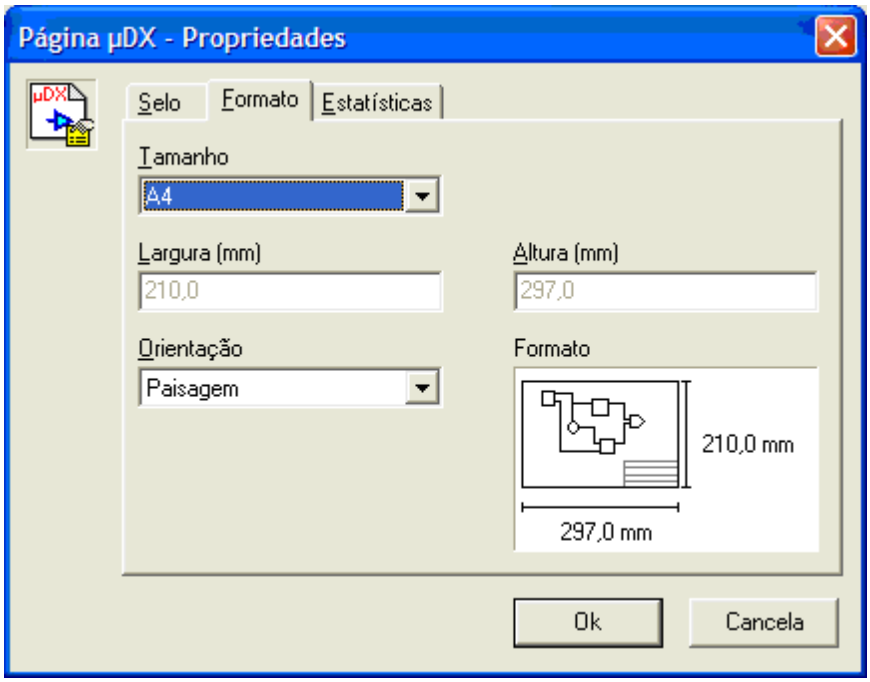
Retorna uma janela com propriedades da página selecionada.

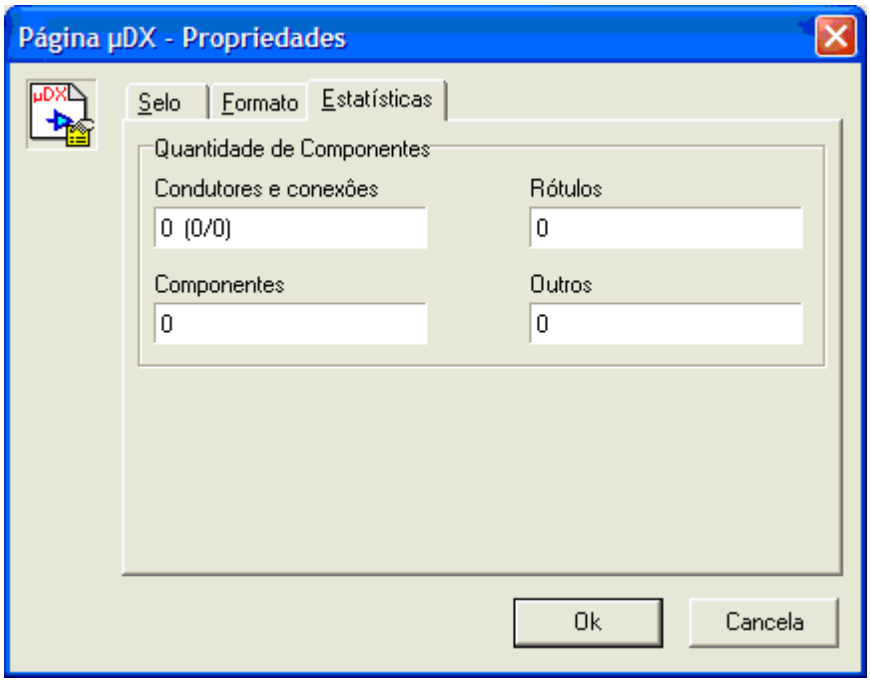
Na aba **Selo** pode-se inserir os dados a serem visualizados no Selo da página (quadro no canto inferior direito com informações sobre a página de programação).

Em **Formato** define-se o tamanho da página e sua orientação.

Por fim, em **Estatísticas** são apresentados alguns dados a respeito da página de programação, como o número de condutores, rótulos, etc., existentes na página.

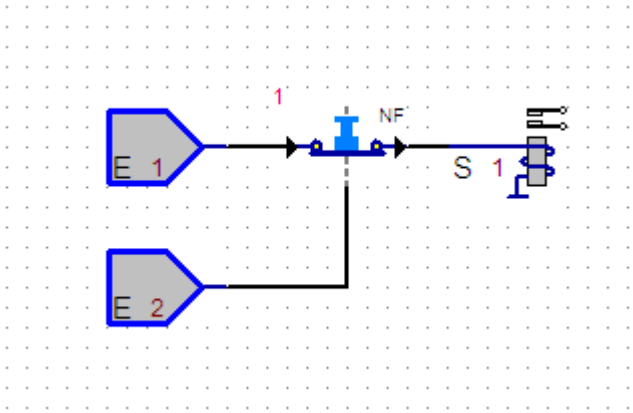



Página µDX - Propriedades
Selo
Formato
Estatísticas
Tamanho
A4
Largura (mm)
210,0
Altura (mm)
297,0
Orientação
Paisagem
Formato
210,0 mm
297,0 mm
Ok
Cancela


Página µDX - Propriedades
Selo
Formato
Estatísticas
Quantidade de Componentes
Condutores e conexões
0 (0/0)
Rótulos
0
Componentes
0
Outros
0
Ok
Cancela

Grade (Ctrl+G)

O posicionamento dos componentes no Editor PG obedecem a um alinhamento em relação a um quadriculado, que pode ser visível ou não, conforme a seleção deste item.



Selo (Shift+Ctrl+S)

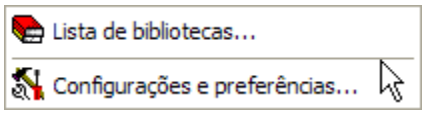
Ativa ou desativa a visualização de selo na página selecionada (página de programação µDX ou página de Macro).

Zoom

Permite selecionar entre sete níveis de ampliação (zoom): 2, 3, 5, 7, 10, 15 ou 20. O nível de zoom pode ser selecionado individualmente para cada página.

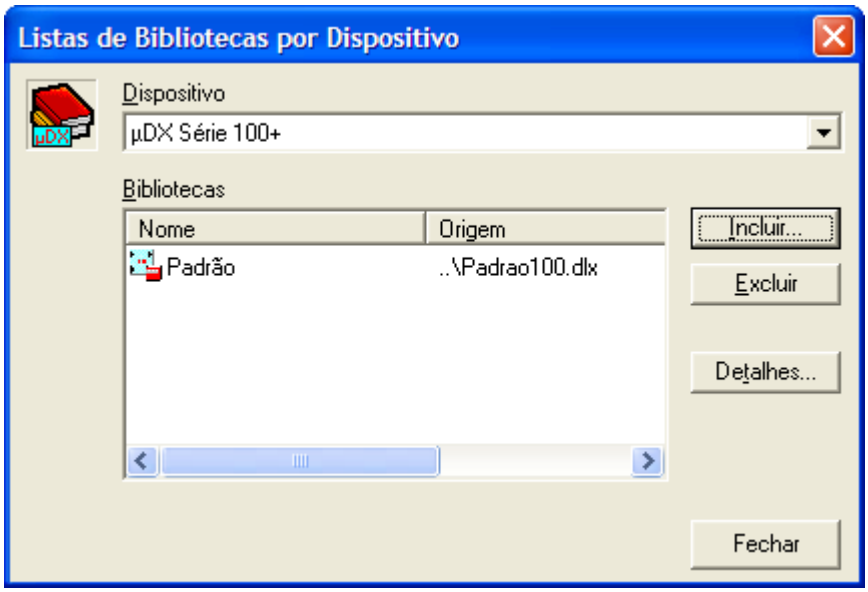
Menu Configurações

Este menu permite escolher as bibliotecas de blocos a serem utilizadas pelo Editor PG, e uma série de configurações para o software.



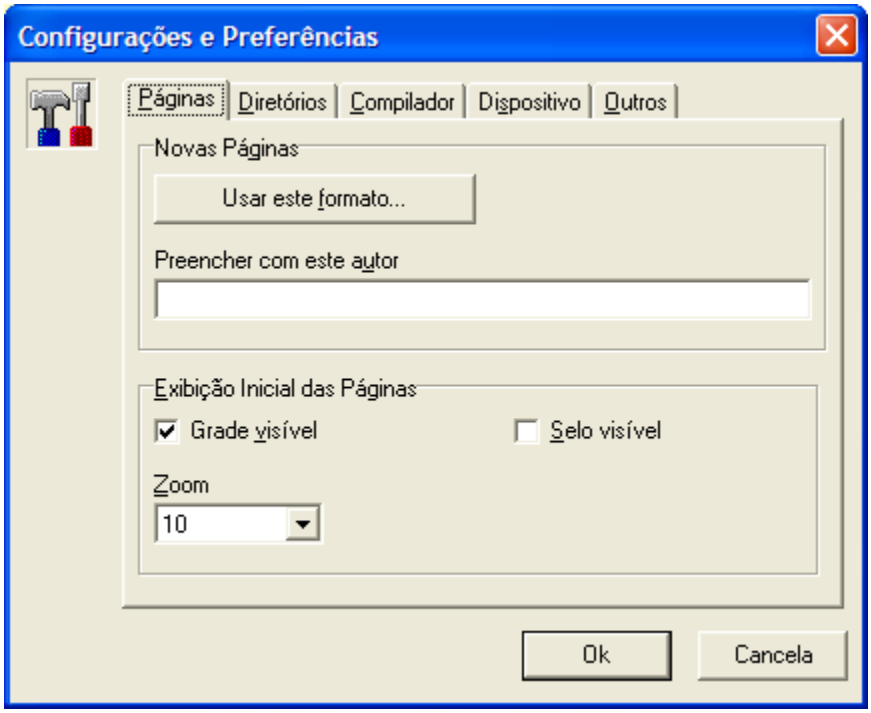
Lista de bibliotecas...

Inclui ou exclui bibliotecas de blocos para o Editor PG. As bibliotecas são arquivos com sufixo .dlx. Acompanha o PG a biblioteca padrão para µDX100 e µDX100+ (Padrao100.dlx).

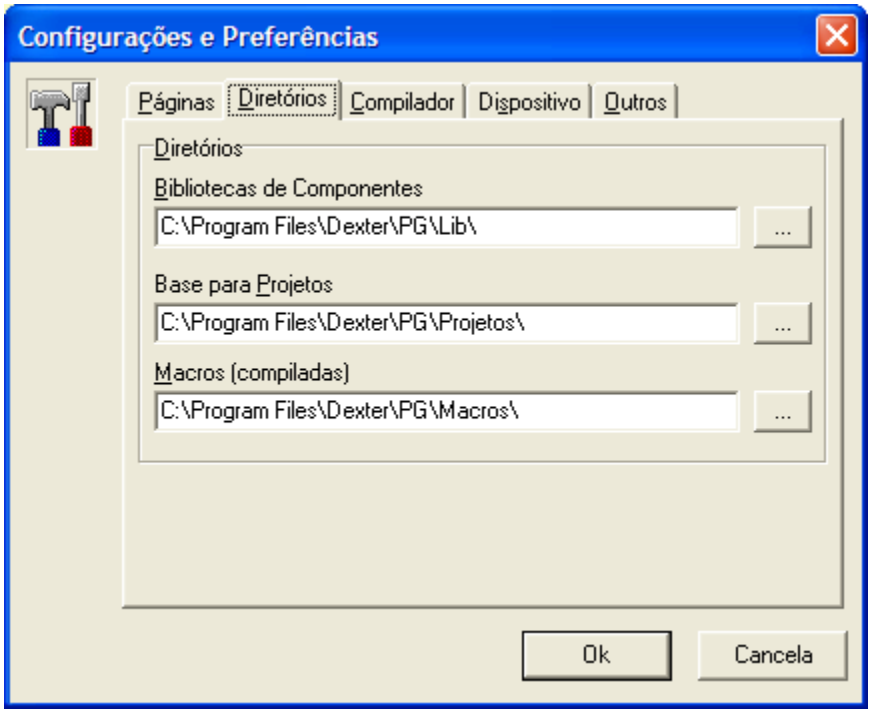


Configurações e preferências...

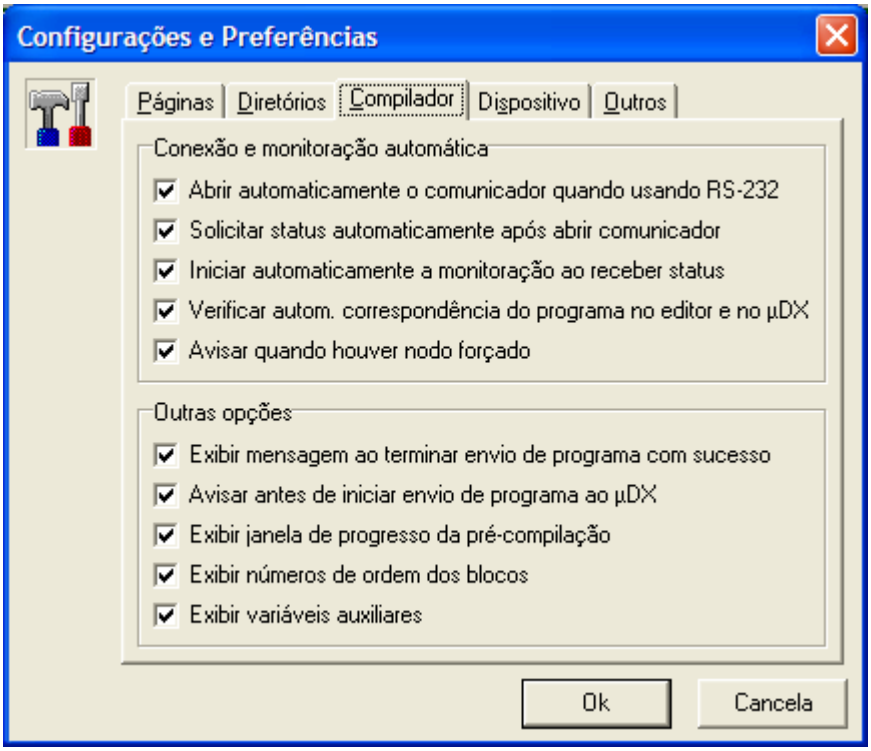
A aba de **Páginas** permite fixar vários parâmetros, como tamanho da página, autor, grade, selo e zoom, que serão usados sempre que forem criadas novas páginas de programação no software PG.



Já a aba de **Diretórios** indica o diretório padrão para as bibliotecas, macros e projetos do Editor PG.

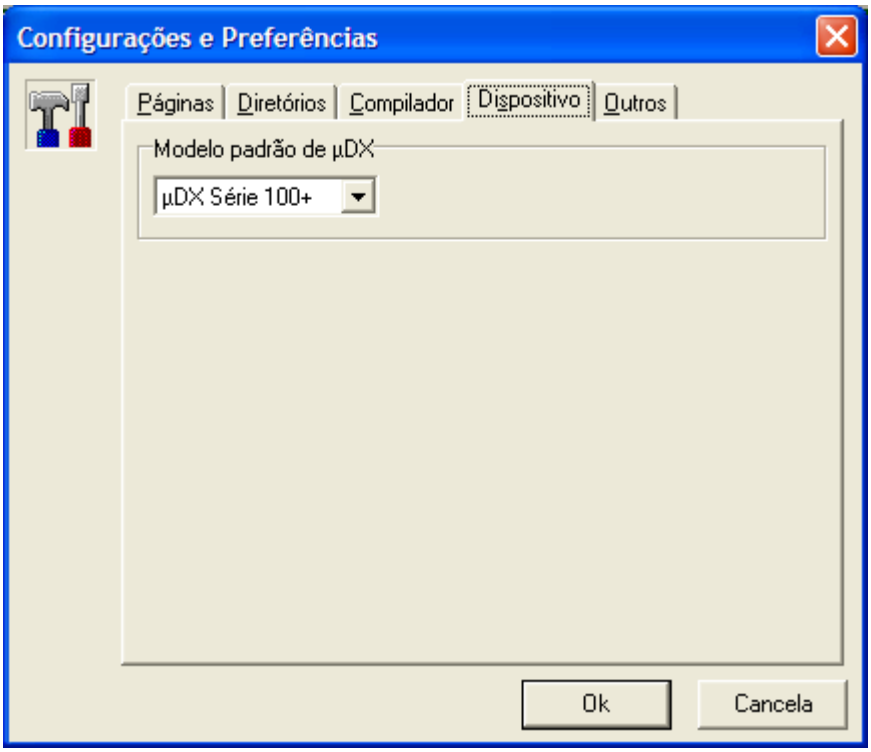


A aba **Compilador** possui diversas opções referentes a janela de compilação:

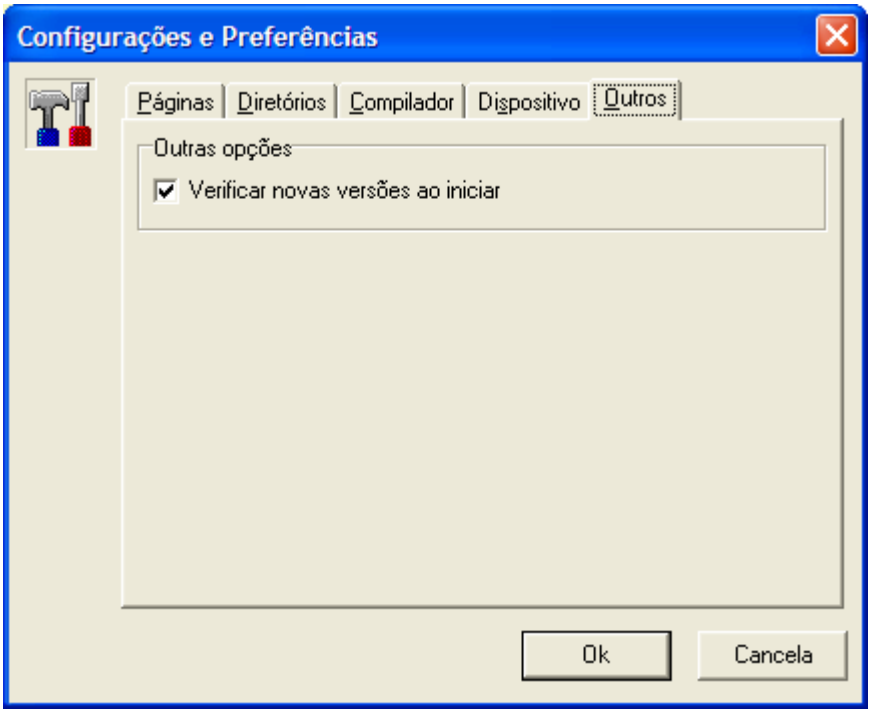


Convém manter marcadas as primeiras quatro opções, de forma que o PG inicie monitoramento automaticamente ao abrir comunicador.

A aba **Dispositivo** permite selecionar qual o dispositivo padrão ao iniciar o software PG. Atualmente é possível escolher entre **µDX100** e **µDX100+**:

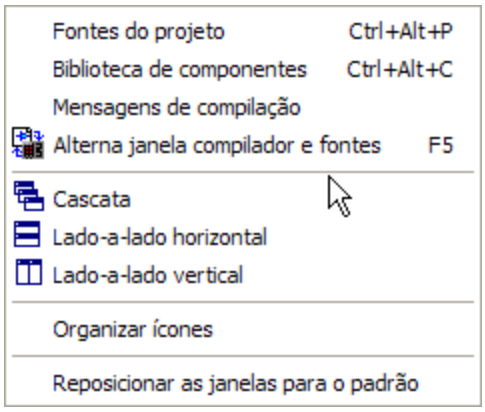


Por fim, a aba **Outros** possui, no momento, apenas a opção de permitir ao software PG verificar no site da Dexter se está disponível uma versão mais atualizada do software. Esta verificação ocorre apenas ao iniciar o software PG, ou selecionando esta verificação no menu Ajuda.



Menu Janelas

Este menu permite migrar entre as várias janelas do Editor PG (fontes, biblioteca, mensagens de compilação), e organizar as janelas de programação na área de programação. Além disso, uma última opção recoloca as janelas em uma posição padrão. As opções não disponíveis são apresentadas em cinza. Note que a opção de **Reposição das janelas para o padrão** não reativa as janelas que foram fechadas. Para isso clique na opção qua ativa a janela correspondente: **Fontes do projeto**, **Biblioteca de componentes**, ou **Mensagens de compilação**.



Fontes do projeto (Ctrl+Alt+P)

Exibe a janela de fontes do projeto, com todas as páginas de programação pertencentes ao projeto.

Biblioteca de componentes (Ctrl+Alt+C)

Exibe a janela de biblioteca de componentes do PG, com todas as famílias de blocos disponíveis.

Mensagens de compilação

Exibe a janela de mensagens de compilação do Editor PG. Note que se trata de uma pré-compilação, que indica variáveis de tipos diferentes interconectadas e outros erros no programa. A compilação propriamente dita ocorre no programa Compilador, que acompanha o pacote de software do controlador µDX100.

Alterna Janela Compilador e Fontes (F5)

Alterna entre janela de Compilador PG e janela com os programas aplicativos elaborados no Editor PG.

Cascata

Organiza as janelas de programação em cascata.

Lado-a-lado horizontal

Distribui as janelas de programação horizontalmente.

Lado-a-lado vertical

Distribui as janelas de programação verticalmente.

Organizar ícones

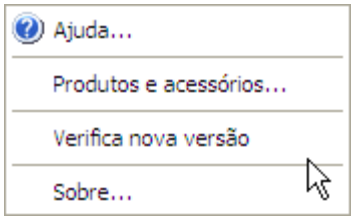
Realinha os ícones das janelas minimizadas no Editor PG.

Reposicionar as janelas para o padrão

Recoloca as janelas principais e flutuantes na configuração padrão do programa.

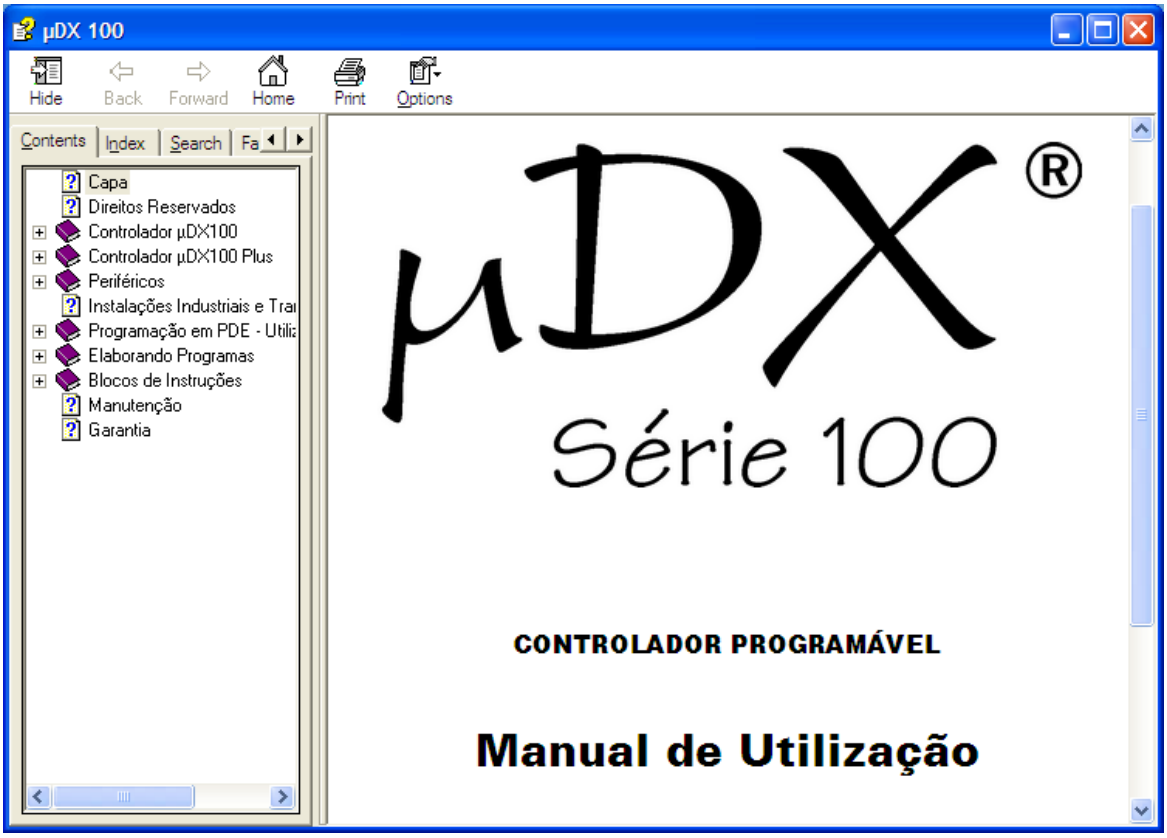
Menu Ajuda

Este menu permite o acesso ao manual (help) do Editor PG. Além disso, a tecla **Sobre...** informa versão do PG e algumas informações do software. Já a opção **Verifica nova versão** faz com quem o PG compare sua versão com a existente no site da Dexter - http://www.dexter.ind.br - e informe se existe uma versão mais atual. A opção **Produtos e Acessórios...** abre um arquivo no formato PDF com descrição de todos os produtos Dexter para linha µDX100.



Ajuda...

Acessa o arquivo de ajuda (help) do Editor PG.

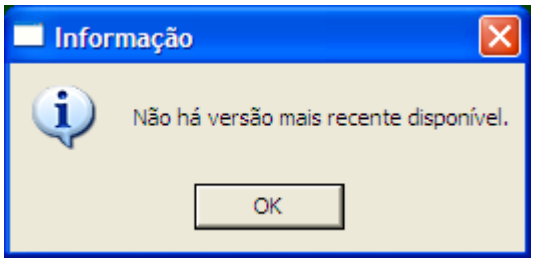


Produtos e Acessórios...

Abre catálogo em formato PDF com descrição de todos os produtos Dexter para linha µDX100.

Verifica nova versão

Verifica se existe uma versão mais atual do software PG no site da Dexter - http://www.dexter.ind.br.

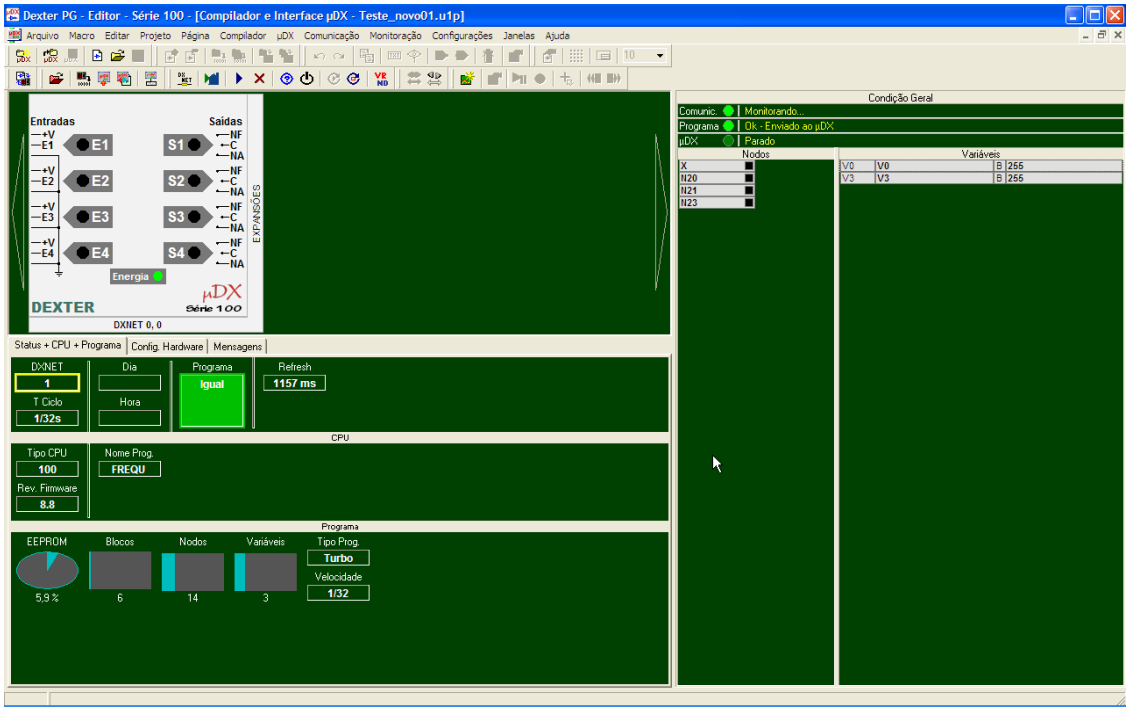


Sobre...

Informações sobre o software Editor PG.

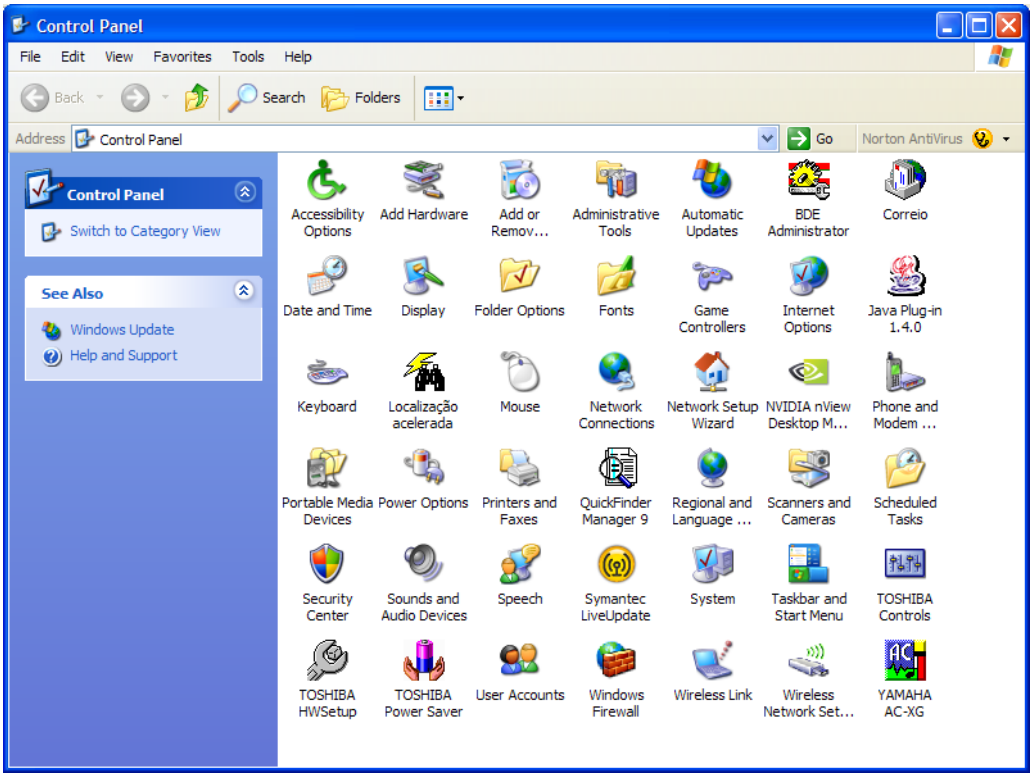
Teclas de Operação do Compilador PG

O Compilador PG (Programador Gráfico) não só permite compilar programas gerados no Editor PG, como monitorar status, nodos e variáveis do Controlador µDX100.

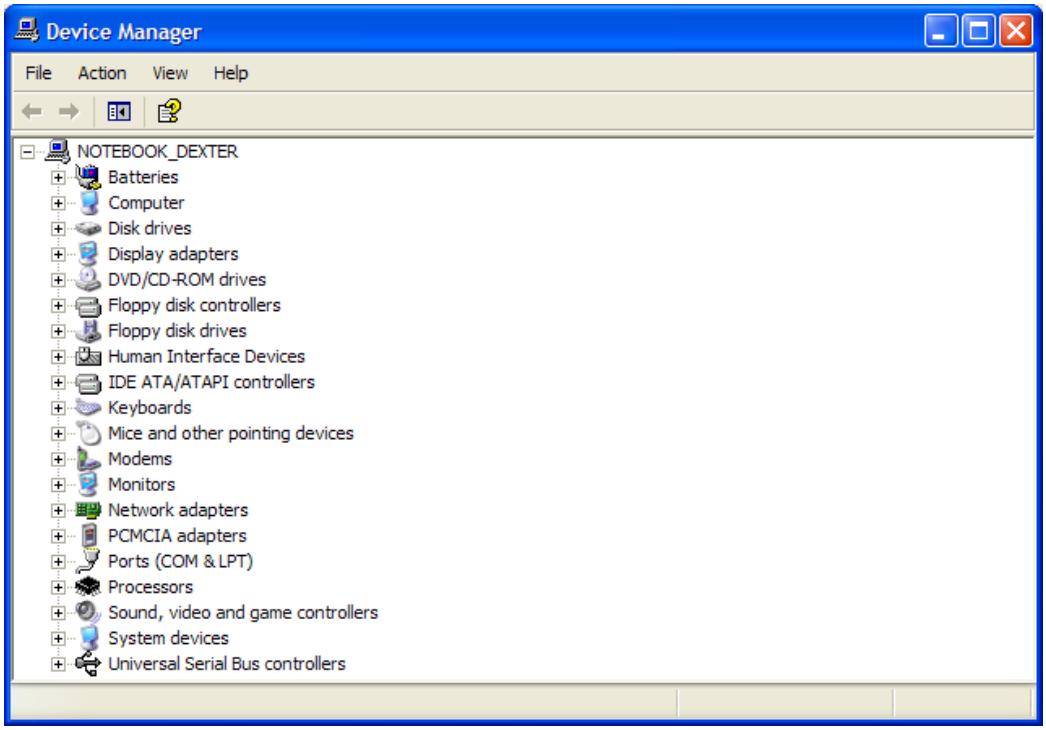


Caso seja estabelecida comunicação serial com o Controlador µDX100 a tela é preenchida com vários dados de status.

Para estabelecer comunicação serial com o µDX100 ligue o cabo serial que acompanha o Modem para µDX100 à porta serial de seu computador (ou a um cabo adaptador USB ⇔ Serial caso seu computador não possua porta serial), e selecione o número de porta serial no Compilador. Para saber quais as portas seriais disponíveis vá no Windows em [Iniciar] → [Configurações] → [Painel de Controle]:



Selecione [Sistema] → [Hardware] → [Gerenciador de Dispositivos]:



A seguir, clique no símbolo [+] do item Portas (COM & LPT) para visualizar as portas seriais (COM) disponíveis. De posse dos endereços das portas seriais retorne ao Compilador PG, selecione o menu **Comunicação** → **Configurar Comunicador**:

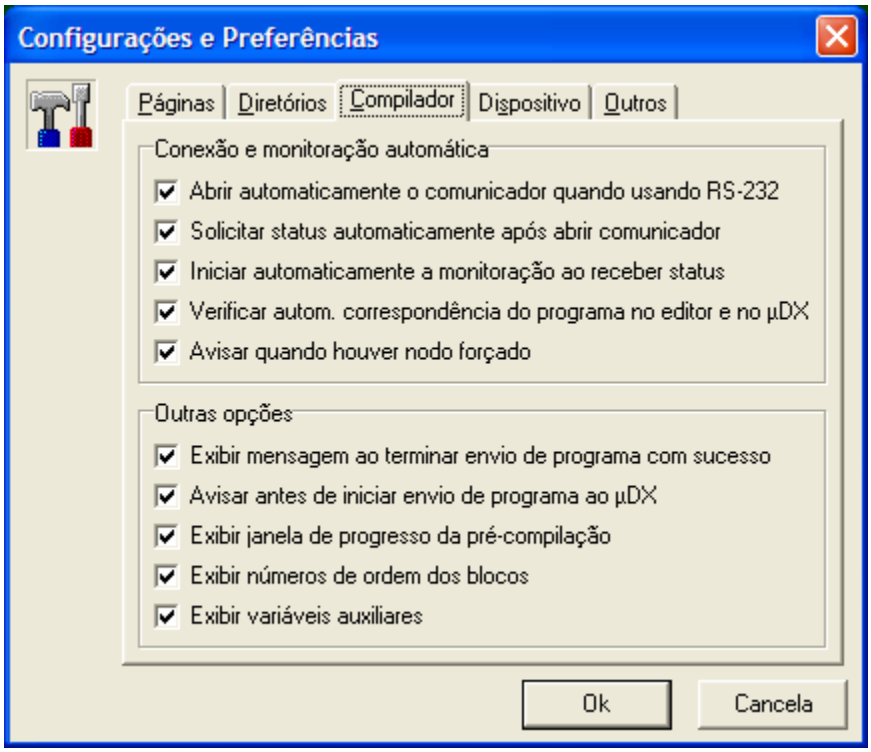
Selecione meio de comunicação RS-232, conforme a figura acima. Escolha o protocolo a ser utilizado (DXNET ou DXNET+) conforme o protocolo programado no MOdem para µDX100. Os Modems são fornecidos de fábrica com protocolo DXNET, 9600 bps, 8 bits, sem paridade, 2 stop bits. A seguir, vá para a aba **RS-232**:

Selecione a porta de comunicação correta e pressione **Ok**. Note que os demais parâmetros não necessitam ser modificados. O Modem para µDX100 é fornecido programado para comunicação serial à 9600bps, 8 bits, sem paridade e 2 stop bits. Por fim, clique no ícone para abrir comunicador  , ou selecione **Comunicação** → **Abrir**.

Se for estabelecida comunicação serial deverão surgir diversos dados do controlador µDX100. Para isso devem estar selecionadas as opções:

[x] Abrir automaticamente o comunicador quando usando RS-232
[x] Solicitar status automaticamente após abrir comunicador
[x] Iniciar automaticamente a monitoração ao receber status
[x] Verificar autom. correspondência do programa no editor e no µDX

Estas opções estão disponíveis no menu **Configurações...** → **Configurações e Preferências** → **Compilador**.



Também é possível executar estes passos manualmente. Para isso clique nos seguintes ícones:

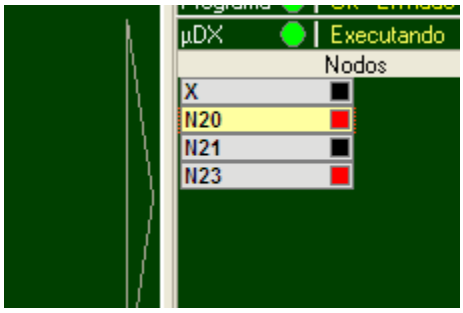
 Requisita status do controlador programável.

 Inicia monitoramento.

Note a existência de vários dados lidos do µDX100 na tela do Compilador. Assim, o Compilador lê do µDX100 as informações de:

- Nome do programa (em 8 caracteres)
- Versão do firmware (software interno do µDX100)
- Tipo de CPU (µDX100 ou µDX100+)
- Tempo de ciclo de execução do programa aplicativo (1/16, 1/32, 1/64 ou 1/256s)
- Hora do relógio de tempo real do µDX100
- Endereço DXNET do controlador µDX100

Além disso, estão disponibilizados todos os nodos utilizados no programa sob a forma de leds (lâmpadas). Estes assumem a cor preta quando desligados, vermelho quando ligados, e vermelhos com moldura amarela quando estão ligados devido ao forçamento via Compilador. Para forçar um destes sinais a nível alto basta clicar duas vezes sobre o led correspondente com a tecla esquerda do mouse.

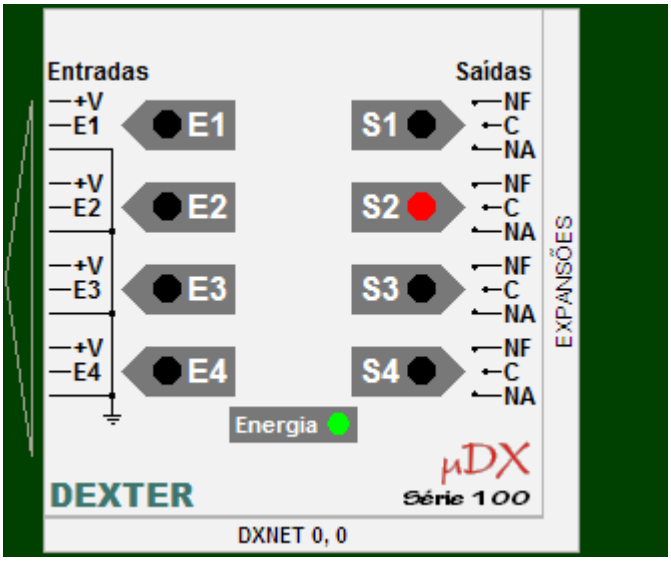


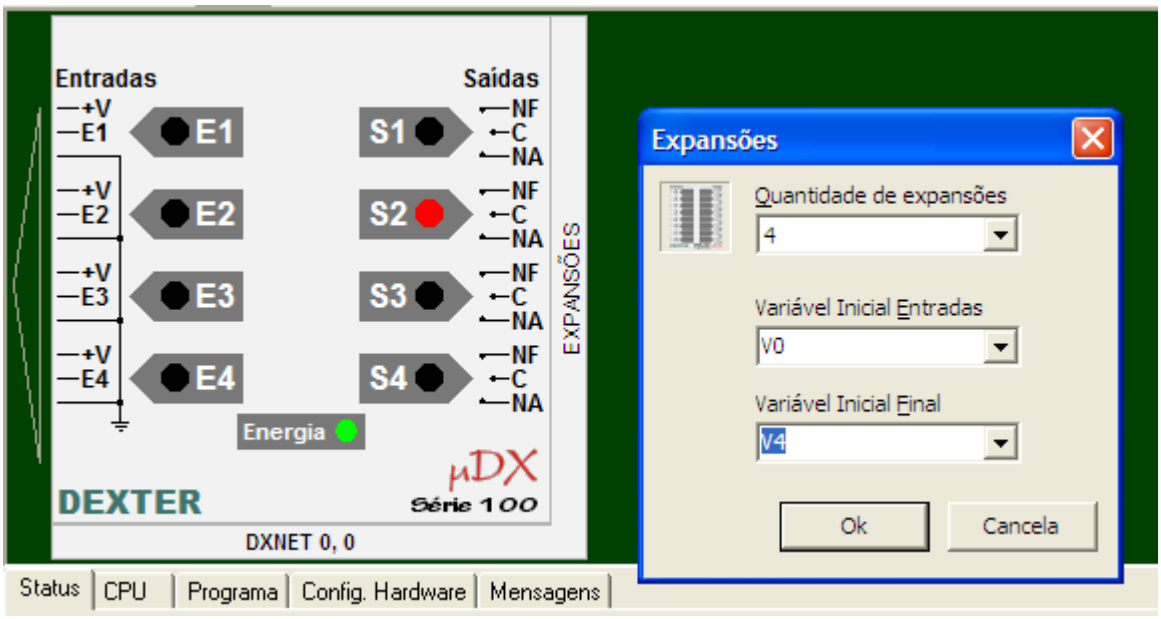
Por fim, ao lado aparecem todas as variáveis usadas no programa aplicativo do µDX100. O tipo de variável é especificada pela letra logo após a designação da variável. Como o µDX100 possui apenas variáveis tipo byte este campo sempre é preenchido com a letra B. Para forçar o valor de uma variável basta clicar com o mouse duas vezes sobre a variável.

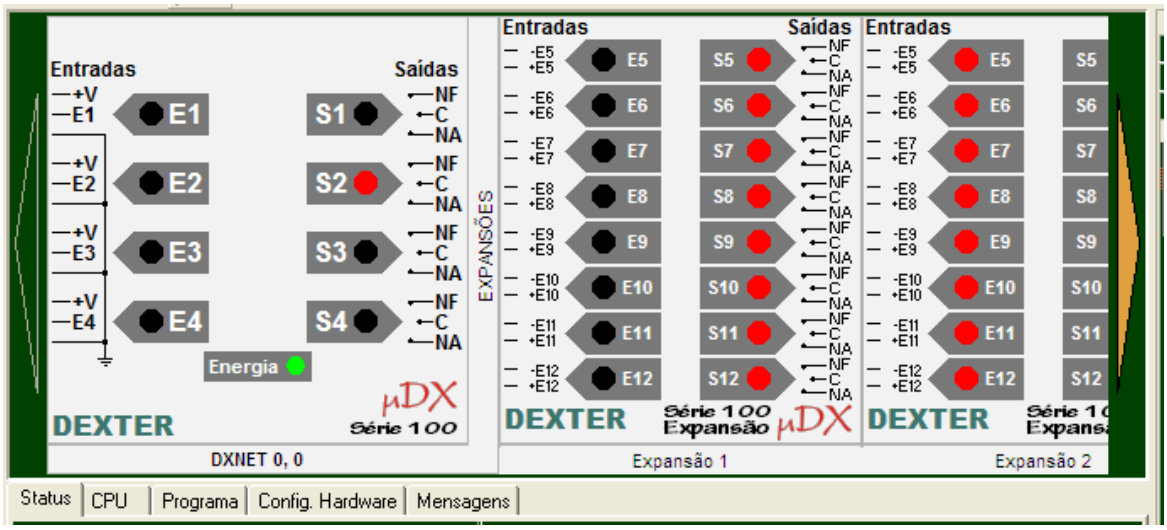
Cabe aqui uma descrição das várias áreas existentes na janela do Compilador PG e suas funcionalidades:

Área do Controlador µDX100 e Expansões

Esta área mostra uma vista do painel frontal do µDX100 e das Expansões atualmente sendo varridas pelo controlador. Note que as Expansões devem ser especificadas pelo usuário para que sejam monitoradas. Para isso basta clicar duas vezes com o mouse sobre a palavra **Expansões** existente à direita da representação do µDX100 (ou selecionar o menu pop-down **µDX** e selecionar **Expansões...**) e informar o número de Expansões, variável inicial de entradas e variável inicial de saídas.



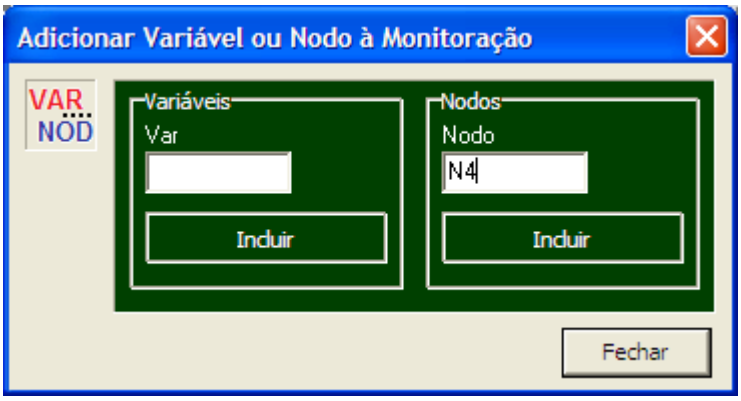




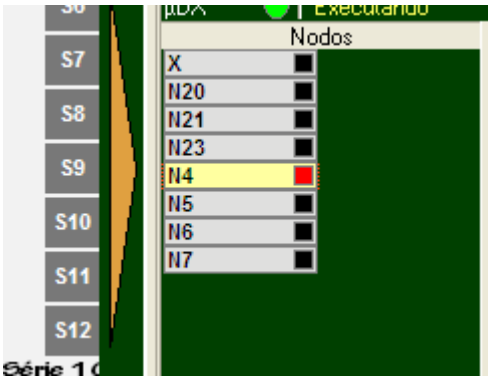
As setas existentes à direita e à esquerda permitem acessar as demais Expansões para µDX100, quando existirem. Neste caso as setas ficam ativas (com cor laranja). Os leds representam o estado das entradas e saídas do controlador e das expansões.

Ao contrário do PG para controlador µDX200, o PG para µDX100 não permite acionar diretamente entradas e saídas do µDX100 e de suas Expansões clicando com o mouse sobre os leds correspondentes. Mas é possível acrescentar nodos a lista de nodos monitorados, assim como acrescentar variáveis a lista de variáveis monitoradas. Com isso, podemos, por exemplo, acrescentar os nodos N4 a N7, correspondentes as saídas S1 a S4 do µDX100. Para isso basta clicar no botão 

existente na barra de ferramentas (ou abrir o menu pop-down **Compilador** e selecionar a opção **Adicionar nodos ou variáveis à monitoração...**) e inserir os nodos e variáveis desejadas.



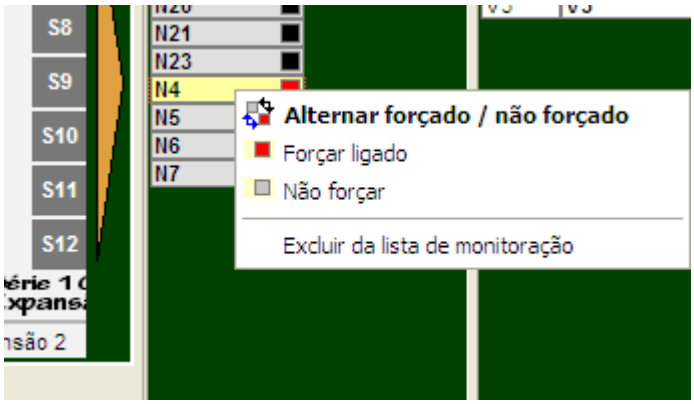
Uma vez incluídos os nodos correspondentes as saídas do controlador basta clicar duas sobre os mesmos para força-los e, conseqüentemente, ligar a saída do µDX100.



Também é possível apontar para o nodo desejado e clicar a tecla esquerda do mouse. Surge uma janela com as seguintes opções:

- **Alternar forçado / não forçado**
- **Forçar ligado**
- **Não forçar**
- **Excluir da lista de monitoração**

Note que esta operação acessa os nodos DXNET do Controlador µDX100. Como estes nodos fazem uma operação OR com os nodos calculados pelo programa aplicativo, sempre é possível forçar a energização de um nodo, mas não seu desligamento (caso o nodo esteja ligado devido ao programa aplicativo não é possível desligá-lo via nodo DXNET; por isso as opções são entre nodo forçado (ligado) ou não forçado (ligado ou não, conforme programa aplicativo)). Abaixo foi usada a tecla direita do mouse sobre o nodo correspondente a saída S1 do controlador µDX100 (nodo N4):



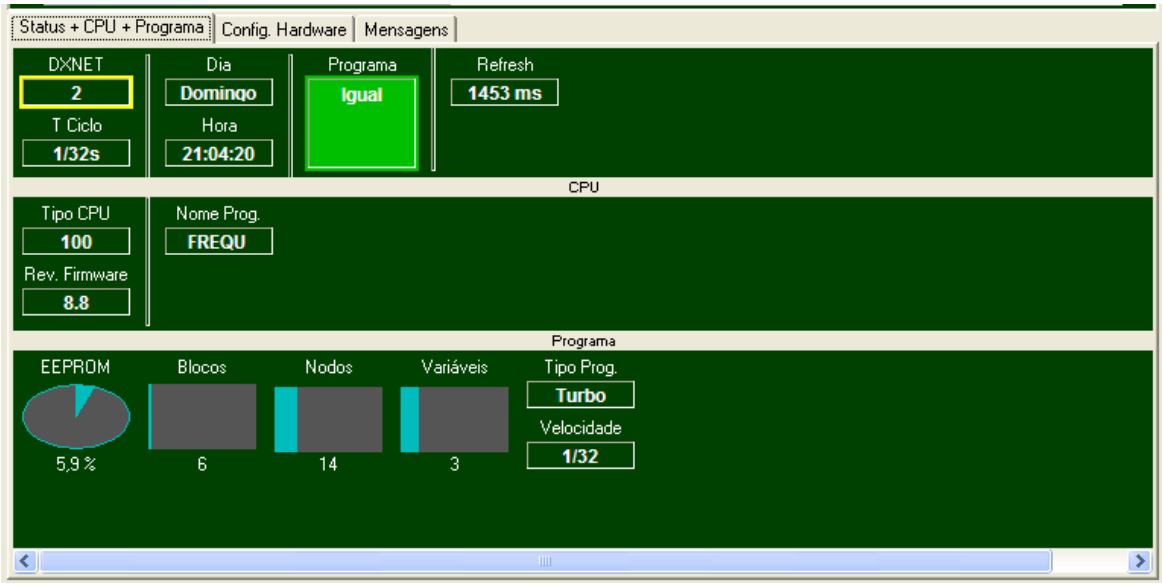
Ainda nesta área é possível editar o endereço e conjunto DXNET acessado pelo Compilador PG. Note que o endereço 0 (zero) e conjunto 0 (zero) sempre acessa o µDX100, independentemente do endereço programado no controlador (endereço broadcast). Ou seja, independentemente do endereço DXNET do µDX100, se for programado no Compilador PG para que ele utilize o endereço 0 (zero) a comunicação será feita com o dispositivo conectado ao computador. O endereço 0 (zero) é o endereço padrão para o PG quando este é instalado, e só deve ser modificado caso existam mais controladores µDX100 ligados via rede DXNET. Para modificar este dado aponte com o mouse para o endereço DXNET (**DXNET 0,0**) que aparece abaixo da representação do µDX100 e clique duas vezes com a tecla esquerda do mouse. Irá surgir a seguinte janela:

Atenção: note que este endereço (conjunto+endereço DXNET) se refere ao endereço que o programa PG irá acessar para transmitir programa aplicativo ou monitorar dados do CLP. Para determinar qual endereço DXNET o controlador irá assumir é preciso acessar a Configuração de Hardware (ver menu pop-down **Compilador**, opção **Configuração de Hardware...**, ou tecla  na barra de ferramentas).

Área de Status do Controlador µDX100 e Programa Aplicativo

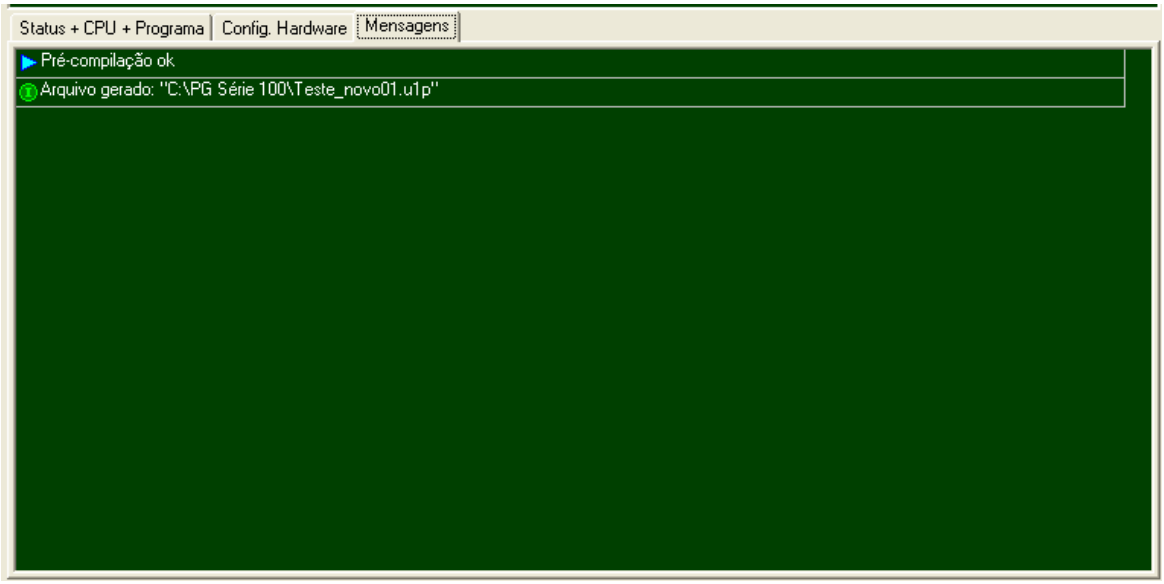
Na primeira aba existem os dados lidos constantemente do µDX100. São eles:

- Endereço DXNET do µDX100.
- Hora e dia da semana do relógio de tempo real do µDX100.
- Tempo de ciclo de execução do programa aplicativo (1/16, 1/32, 1/64 ou 1/256s).
- Status de verificação se programa carregado no PG corresponde ao lido no µDX100.
- Tempo de Refresh da tela do Compilador.
- Tipo de CPU (µDX100 ou µDX100+).
- Revisão de firmware do µDX100.
- Nome do programa aplicativo existente no µDX100 (nome truncado em 7 caracteres).
- Ocupação de memória EEPROM do programa aplicativo carregado no Compilador PG, em percentual.
- Ocupação de memória RAM do programa aplicativo carregado no Compilador PG, em percentual.
- Número de blocos do programa aplicativo carregado no Compilador PG.
- Número de Nodos do programa aplicativo carregado no Compilador PG.
- Número de variáveis do programa aplicativo carregado no Compilador PG.



Já a segunda aba mostra as configurações de hardware (arquivo sufixo .u1i) pertencentes ao programa aplicativo (arquivo sufixo .u1p) carregado no Compilador PG. Estas configurações determinam o endereço DXNET (conjunto,endereço) e o nome do programa aplicativo (nome este a ser gravado no µDX100).

A terceira e última aba apresenta mensagens de compilação do programa sufixo U1P carregado no Compilador PG:



Área de Condição Geral

Nesta área são apresentados três leds, que indicam o status da comunicação entre computador e µDX100, status do programa aplicativo carregado no Compilador PG, e status do próprio Controlador µDX100. Os leds assumem diferentes cores, conforme a seguinte convenção:

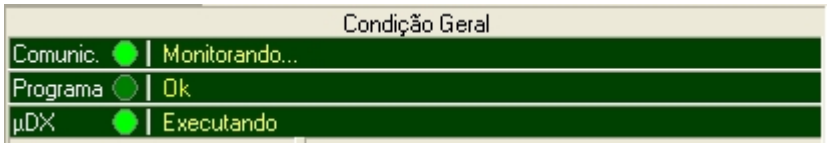
Comunicação:

Cinza → Comunicador Parado (sem comunicação).
Verde Escuro → Comunicador Aberto (sem comunicação).
Verde Claro → Monitoramento ativo (comunicação constante com µDX100).
Laranja → Comunicando (comunicação temporária de comando com µDX100).
Vermelho → Erro de comunicação (Timeout, erro de CRC).

Vermelho/Amarelo → Comunicação para carga de programa aplicativo.

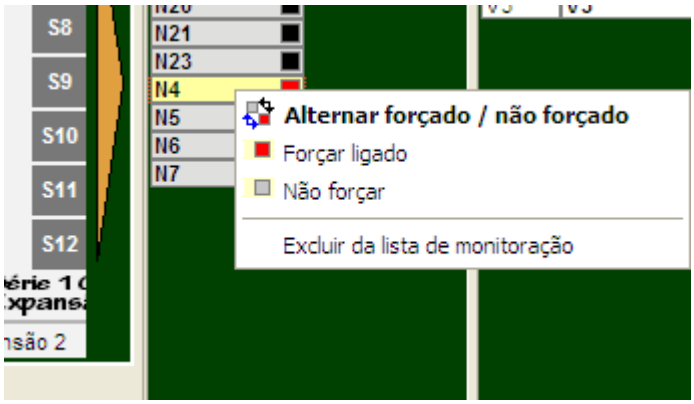
Programa:
Cinza → Nenhum programa aplicativo carregado no Compilador PG (arquivo sufixo .u1p).
Verde Escuro → Programa carregado no Compilador PG e compilado sem erros.
Verde Claro → Programa transmitido para µDX100 com sucesso.
Vermelho → Erro ao compilar programa aplicativo.

µDX:
Cinza → Não foi lido status do µDX100.
Verde Escuro → Programa aplicativo parado no µDX100.
Verde Claro → Programa aplicativo executando no µDX100.
Vermelho → Erro no µDX100.

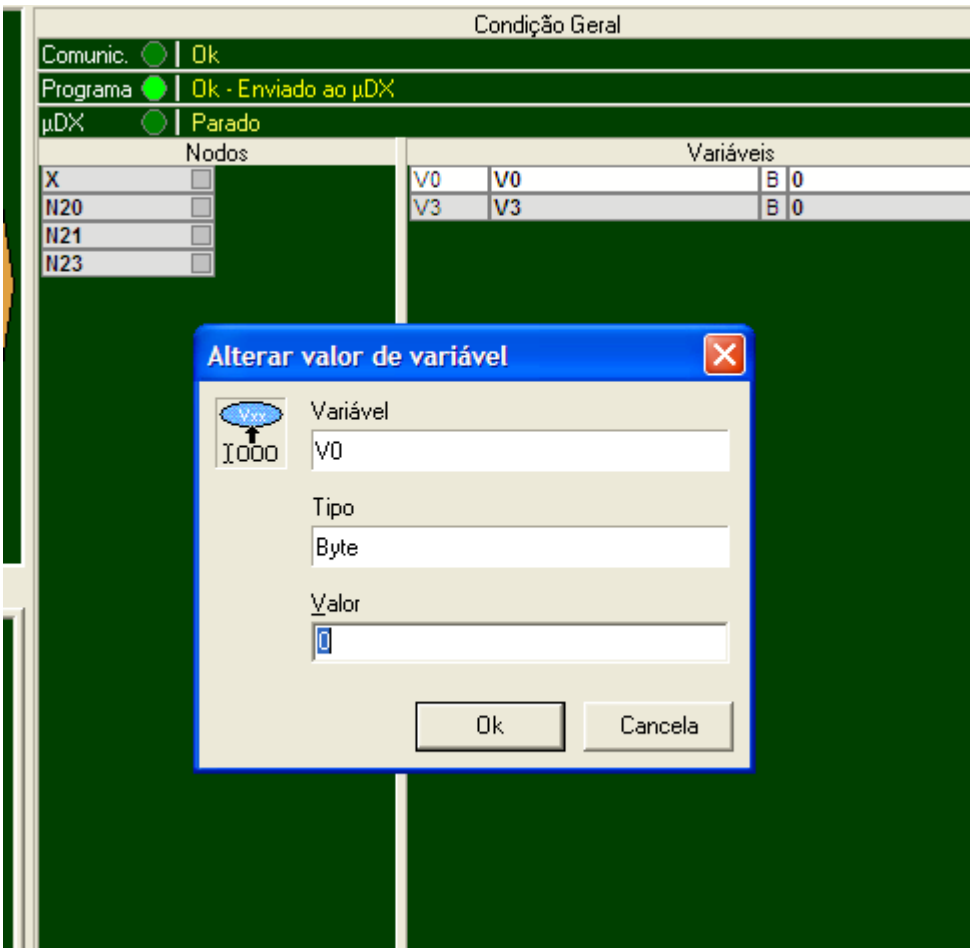


Área de Nodos e Variáveis do Controlador µDX100

A área de nodos disponibiliza os nodos utilizados no programa aplicativo (para isso o programa aplicativo carregado no Compilador PG deve ser o mesmo que está no µDX100). É possível forçar nodos clicando duas vezes com a tecla esquerda do mouse com o cursor sobre o nodo desejado. Com a tecla direita se abre um menu de opções, como representado a seguir:



Já a área de variáveis apresenta variáveis utilizadas no programa aplicativo (desde que o programa aplicativo carregado no Compilador PG seja o mesmo que está no µDX100. É possível forçar valores nestas variáveis. Para isso basta apontar com o cursor para a variável e clicar duas vezes com a tecla esquerda do mouse:

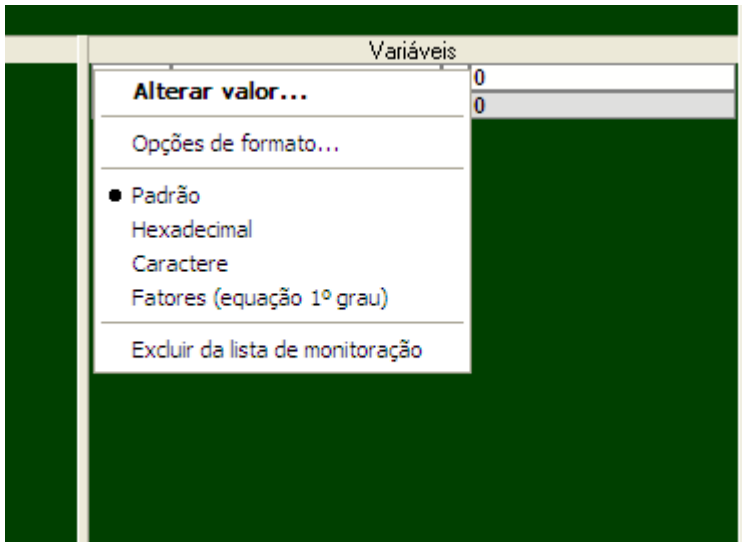


O exemplo anterior mostra o forçamento de valor na variável v0.

Note que as variáveis possuem quatro colunas. A primeira indica seu endereço absoluto, a segunda indica o nome da variável, a terceira coluna indica o tipo de variável que no caso de µDX100 ou µDX100+ é sempre byte - B), e a quarta coluna mostra o valor da variável.

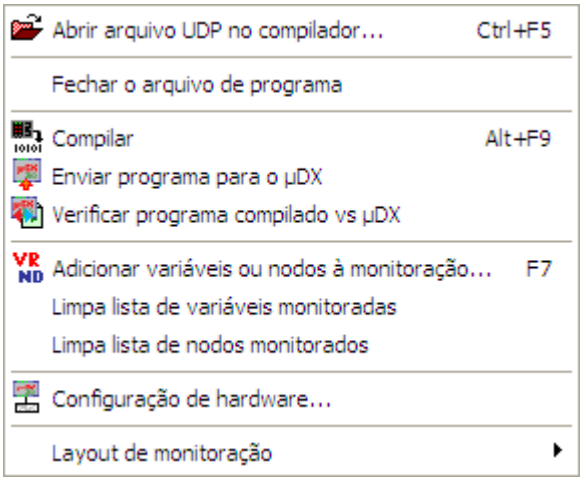
Clicando sobre a variável com a tecla direita do mouse surge uma janela de opções como mostrado a seguir. Nela existem as seguintes possibilidades:

- **Alterar valor:** exatamente a mesma funcionalidade já descrita obtida ao clicar duas vezes sobre a variável.
- **Opções de formato:** permite escolher entre notação Padrão, Hexadecimal, Caracter ASCII, Expoente (notação de engenharia), ou ainda com Fatores (equação de primeiro grau y=ax+b). Esta última opção permite converter a variável para outro valor. Por exemplo, se a variável representa uma temperatura pode-se converter o valor da variável para que apresente a temperatura diretamente em graus celsius. As opções de formato estão disponíveis também diretamente na janela, de forma a permitir comutar rapidamente entre as representações.
- **Excluir da lista de monitoração:** permite retirar esta variável do monitoramento, tornando o ciclo de varredura mais rápido.


Variáveis
0
0
Alterar valor...
Opções de formato...
Padrão
Hexadecimal
Caractere
Fatores (equação 1º grau)
Excluir da lista de monitoração

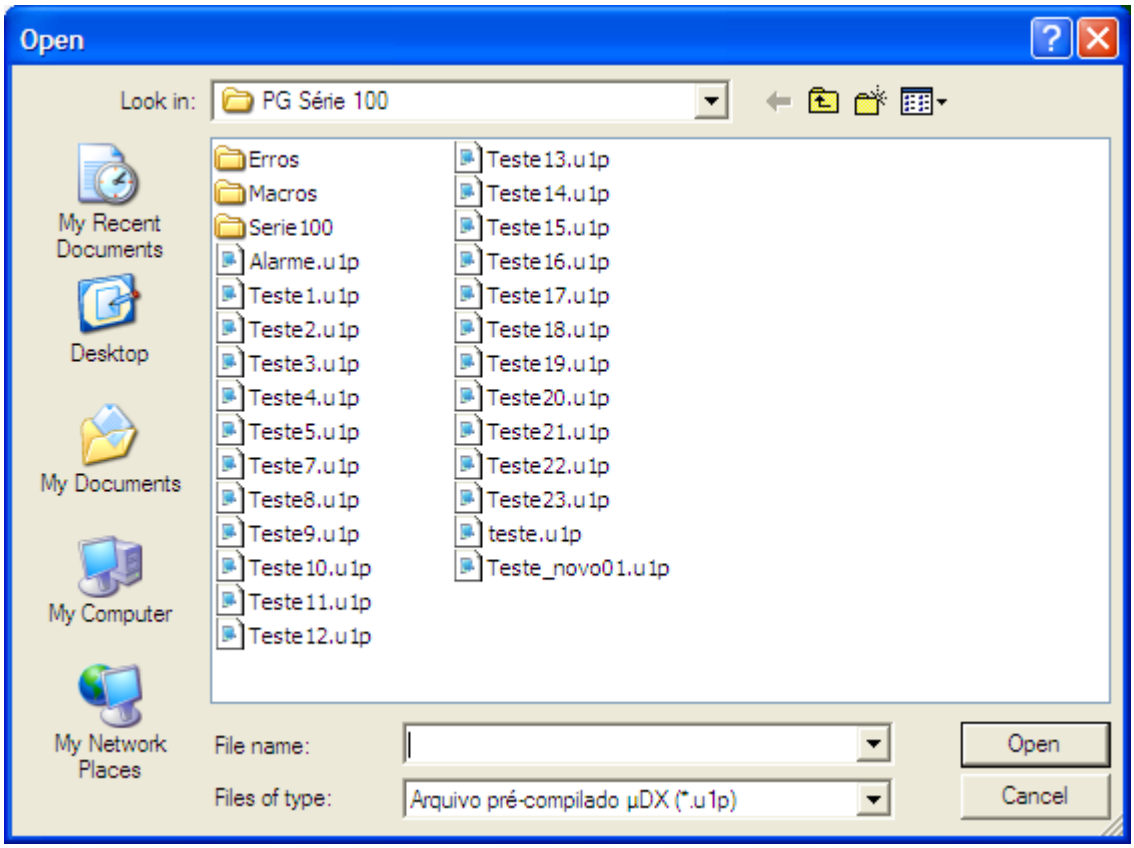
Menu Compilador

O menu Compilador possibilita carregar programas, compilá-los, configurar o hardware associando esta configuração ao programa caregado no Compilador, e abrir janela de log. Também é possível editar um layout de monitoração, escolhendo quais as variáveis e nodos do programa aplicativo devem ser monitorados.



Abrir Arquivo UDP no Compilador... (Ctrl+F5)

Permite abrir no Compilador PG o programa pré-compilado no Editor PG. Note que uma página de programação (sufixo .d1g) ou um projeto (sufixo .d1p) é pré-compilado no Editor PG, gerando um arquivo sufixo .u1p, que pode ser lido no Compilador (que irá efetuar a compilação final, permitindo a transmissão do programa aplicativo para o µDX100). Ao selecionar esta opção surge a janela a seguir, que permite a escolha do programa a ser carregado:



Fechar o Arquivo de Programa

Fecha o programa aplicativo pré-compilado (sufixo .u1p) previamente aberto no Compilador.

Compilar (Alt+F9)

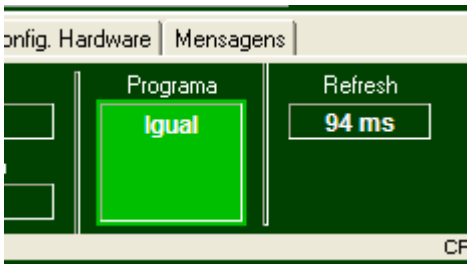
Compila o programa previamente carregado no Compilador. Note que, ao carregar um programa sufixo .u1p no Compilador PG ele já é compilado. Assim, esta tecla é útil apenas para efetuar nova compilação (por exemplo, para verificar as mensagens de compilação no log do compilador).

Enviar Programa para o µDX

Envia o programa aplicativo compilado para o µDX100. Para isso é necessário que o canal de comunicação esteja aberto (Comunicador aberto).

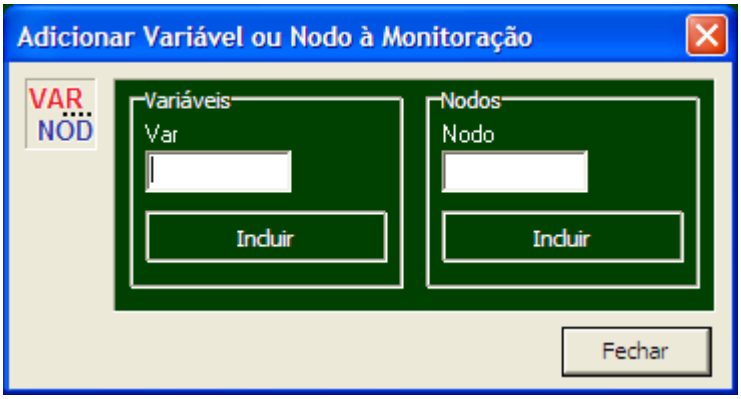
Verificar Programa Compilado versus µDX

Verifica se o programa compilado existente no software PG é idêntico ao existente no controlador µDX100. Se forem idênticos a área de verificação de programa fica em verde e é iniciado o monitoramento de variáveis e nodos do programa aplicativo.



Adicionar nodos ou variáveis à monitoração... (F7)

Permite acrescentar variáveis e nodos à monitoração. Note que apenas nodos e variáveis devidamente nomeados no programa aplicativo são monitorados. Com este recurso qualquer outro nodo ou variável pode ser monitorado.



Limpa lista de variáveis monitoradas

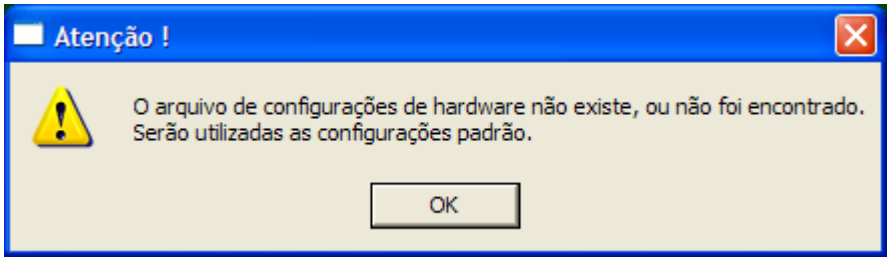
Retira todas as variáveis da monitoração.

Limpa lista de nodos monitorados

Retira todos os nodos da monitoração.

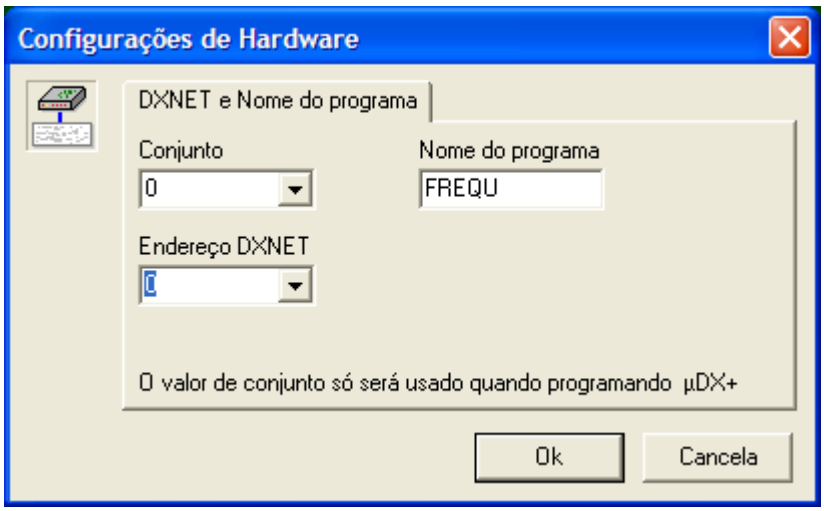
Configuração de Hardware

Ao carregar pela primeira vez um programa aplicativo pré-compilado (sufixo .u1p) no Compilador PG aparece a seguinte mensagem de advertência:



Isso porque não havia sido gerado ainda um arquivo de configuração de hardware (sufixo .u1i) associado a este programa aplicativo. O arquivo de configuração de hardware possui o mesmo nome do programa aplicativo, mas com sufixo U1I. Estes parâmetros são associados ao programa aplicativo (arquivo sufixo .U1P) gerado pelo Editor PG. Assim, os parâmetros

escolhidos para determinado programa aplicativo são salvos com o mesmo nome do programa (arquivo sufixo .U1I). O Compilador irá gerar um arquivo padrão de configuração. Esta opção permite editar este arquivo, de forma a modificar parâmetros de hardware associados a este programa (endereço DXNET e nome do programa):



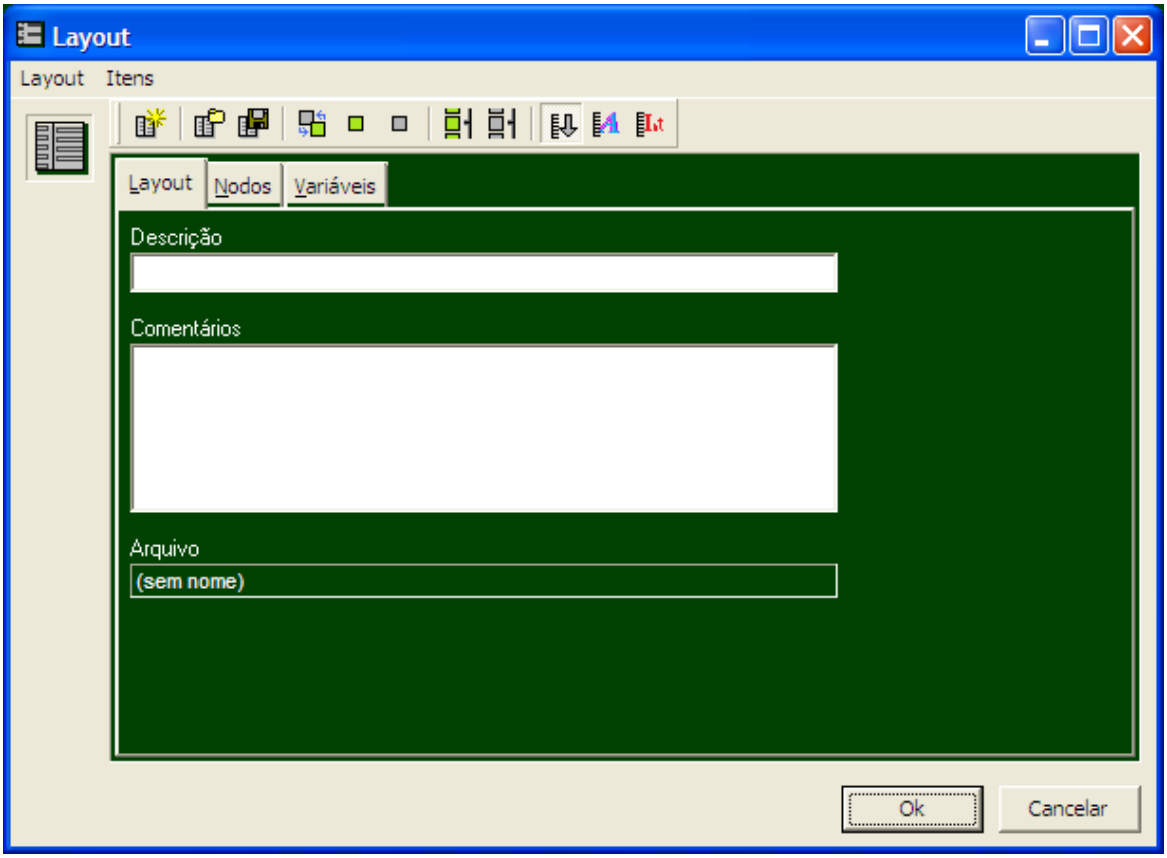
Atenção: O endereço DXNET 0 (zero) no conjunto 0 (zero) é o endereço broadcast, ou seja, todos os controladores µDX100 respondem a comunicações neste endereço, independentemente de seus endereços. Assim, se houver apenas um controlador µDX100 na rede DXNET pode-se programar o CLP com este endereço (0,0). Já se houverem mais controladores na rede DXNET é mandatório programar cada um dos CLPs com endereços distintos e diferentes de 0,0.

Layout de Monitoração

Esta opção permite criar e editar um layout de monitoração. O layout de monitoração é útil principalmente em programas aplicativos extensos, pois permite selecionar quais variáveis e nodos serão monitorados pelo PG, além de classificá-los por ordem alfabética. O layout gerado pode ser salvo (arquivo sufixo **.UDG**) para seu uso posterior. Existem as seguintes alternativas na opção de Layout de Monitoração:

Novo Layout

Permite criar um novo layout de monitoração. Ao selecionar surge a janela de layout de monitoração, conforme mostrado a seguir:



A primeira aba - **Layout** - possibilita inserir uma descrição para o layout de monitoração, e também comentários a respeito do mesmo. Além disso, mostra o nome do arquivo em que o layout foi salvo.

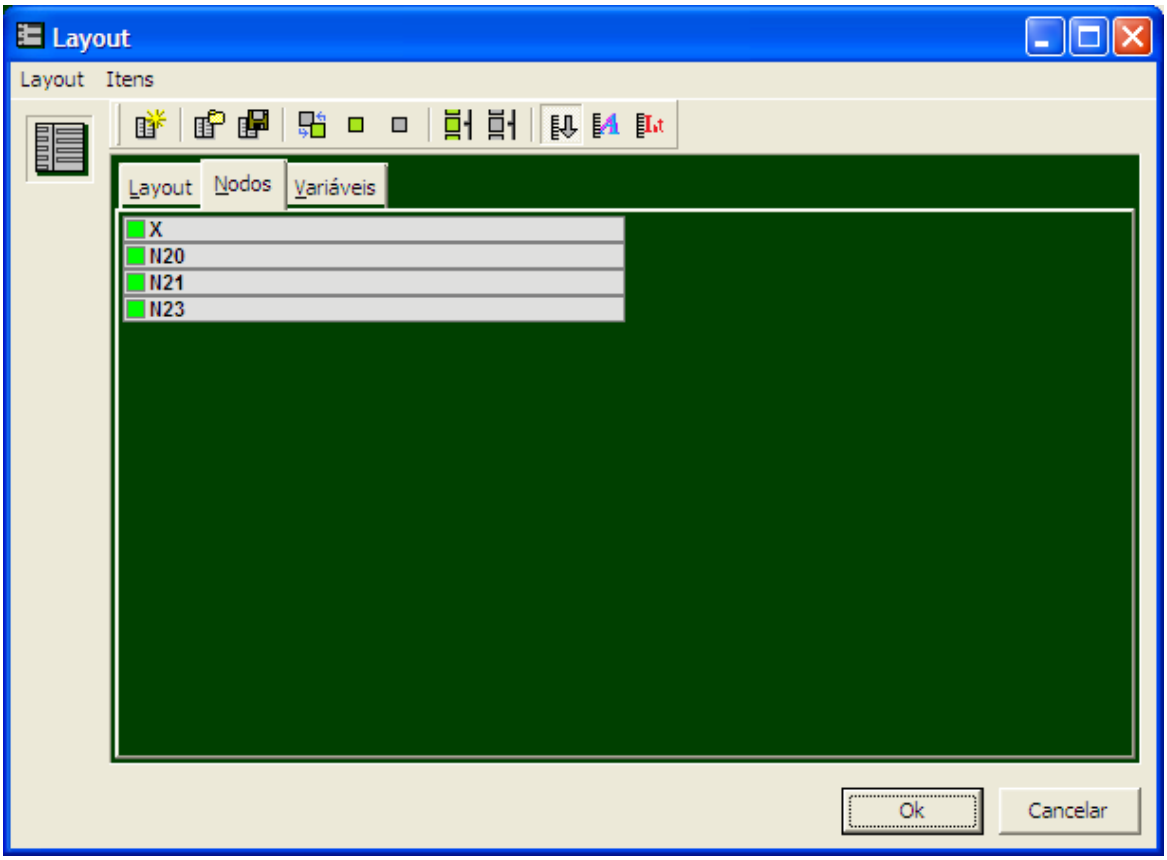
Já os ícones na parte superior desta janela permitem diversas operações. Neste momento vamos nos ater as três primeiras operações, ou seja:

Novo Layout: Zera todas as informações já inseridas no layout, gerando uma janela de layout de monitoração limpa.

Carrega Layout: Permite carregar um layout de monitoração previamente salvo em disco (arquivos sufixo **.UDG**).

Salva Layout: Salva o layout de monitoração em disco (arquivos sufixo **.UDG**).

A segunda aba - **Nodos** - possibilita selecionar quais nodos do programa aplicativo carregado no Compilador PG devem ser monitorados. Todos os itens marcados em verde serão monitorados. Já os itens em cinza serão excluídos, aliviando o processo de monitoração.



Os ícones pertinentes existentes na parte superior da janela são:

Alterna Estado do Item: alterna o item selecionado entre visível e invisível.

Item Visível: torna o item selecionado visível.

Item Invisível: torna o item selecionado invisível.

Todos Visíveis: torna todos os itens visíveis.

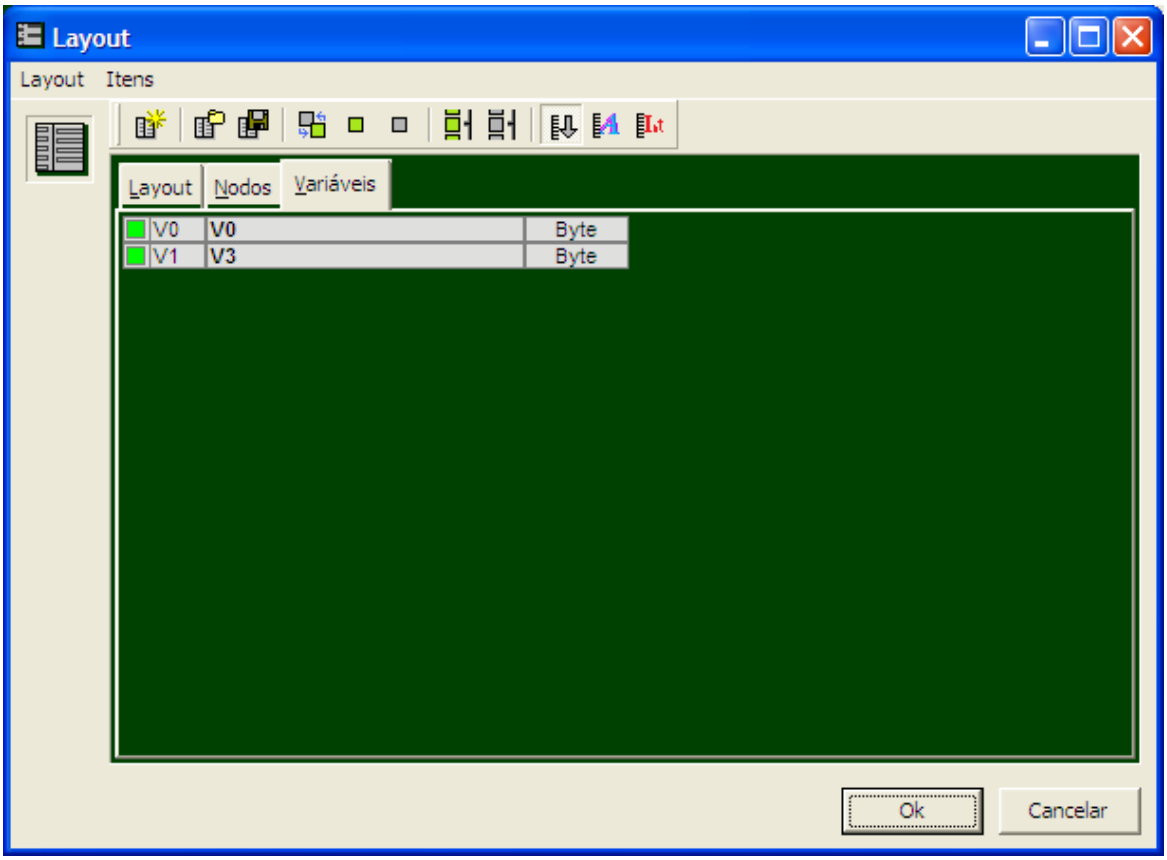
Todos Invisíveis: torna todos os itens invisíveis.

Ordem Padrão: ordena itens pela ordem padrão (variáveis absolutas em ordem crescente).

Ordem Alfabética: ordena itens por ordem alfabética crescente.

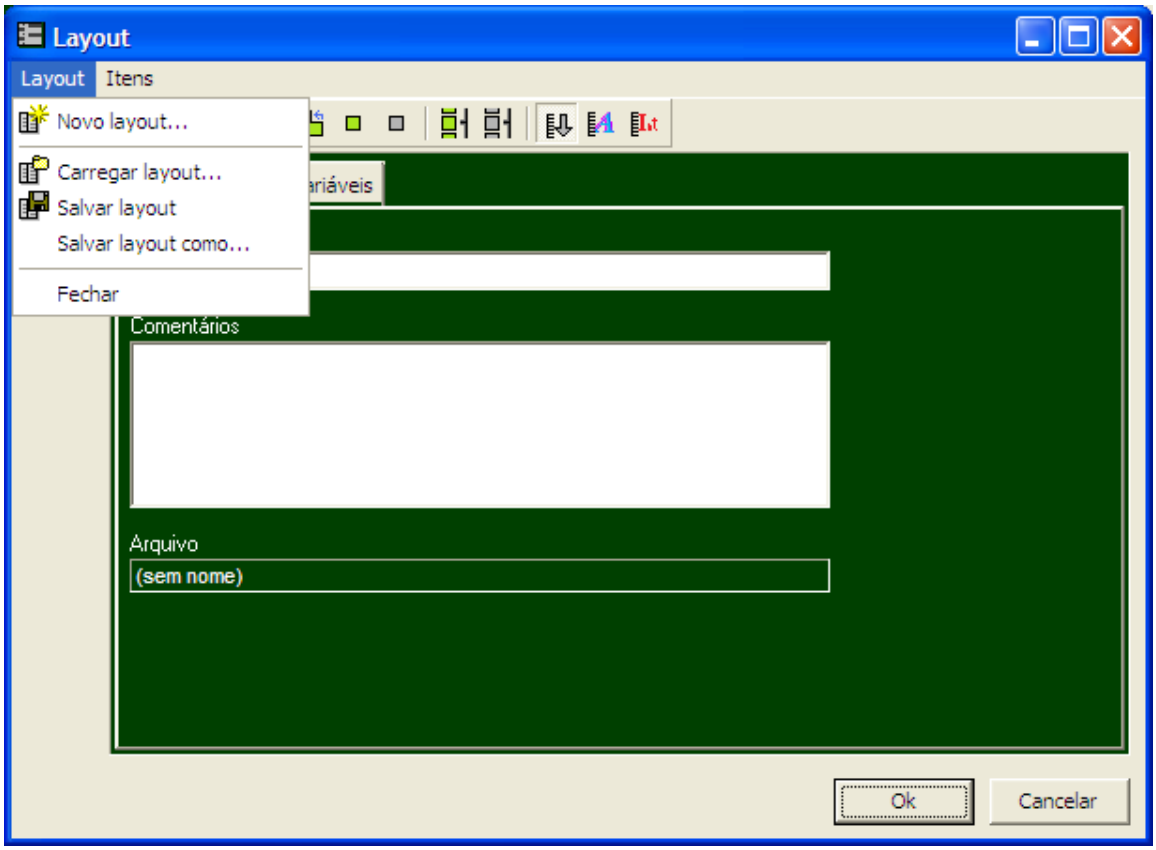
Ordem de Tipo: ordena itens por tipo (no caso só existe tipo nodo e tipo variável byte, de forma que esta tecla é inoperante)

A terceira e última aba - **Variáveis** - apresenta a lista de todas as variáveis utilizadas no programa aplicativo do µDX100:



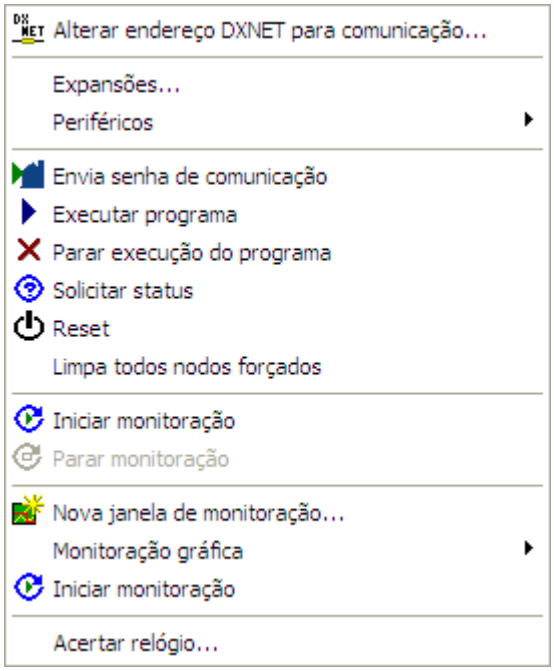
Dica: também é possível alternar entre visível e invisível qualquer item da lista de variáveis ou nodos simplesmente clicando duas vezes com o mouse sobre o item desejado.

Além dos ícones a janela de Layout de Monitoração apresenta dois menus pop-down - **Layout** e **Itens**, com funções idênticas às dos ícones e outras funções de funcionalidade óbvia, como Salvar como...:


Layout
Layout
Itens
Novo layout...
Carregar layout...
Salvar layout
Salvar layout como...
Fechar
ariáveis
Comentários
Arquivo
(sem nome)
Ok
Cancelar

Menu µDX

O Menu µDX permite interagir com o controlador programável, ordenando a execução do programa aplicativo, efetuando reset, solicitando status, etc. Também permite iniciar monitoração gráfica (existe um menu pop-down específico para monitoração gráfica que será descrito adiante, mas muitas de suas funcionalidades estão disponíveis aqui também).



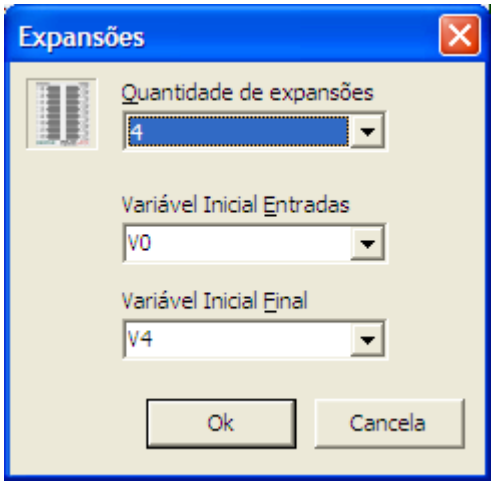
Alterar endereço DXNET para comunicação...

Esta opção permite editar o endereço DXNET acessado pelo Compilador PG. Note que o endereço 0 (zero) conjunto 0 (zero) é o endereço broadcast, ou seja, qualquer µDX100 conectado irá responder a este endereço, independentemente de seu próprio endereço. Este endereço só pode ser usado no caso de apenas um controlador estar conectado à rede DXNET. O endereço 0,0 (zero) é o endereço padrão para o PG quando este é instalado, e só deve ser modificado caso existam mais controladores µDX100 ligados via rede DXNET. Para modificar este dado é possível, além da escolha desta opção no menu µDX, apontar com o mouse para o endereço DXNET (**DXNET 0,0**) que aparece abaixo da representação do µDX100 e clicar duas vezes com a tecla esquerda do mouse. Irá surgir a seguinte janela:

Digite o valor desejado e clique em **Ok**. Note que no caso de controlador µDX100 (em vez de µDX100 Plus) não existe a opção de conjunto.

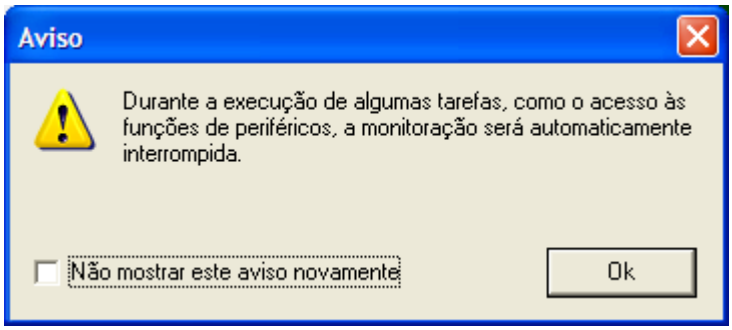
Expansões...

Permite especificar ao monitoramento do Compilador PG a quantidade de Expansões de Entradas/Saídas presentes e quais as variáveis iniciais de entrada e de saída. Note que, no caso de µDX100 Turbo só é permitida uma Expansão. Já no caso de µDX100 Plus são permitidas até quatro Expansões. Deve-se especificar a variável inicial para entradas e a variável inicial para saídas, conforme determinado no programa aplicativo. O monitoramento irá alocar as variáveis seqüencialmente, de forma similar ao controlador. Por exemplo, se especificarmos v0 como variável de entrada e v4 como variável de saída, e houver 4 Expansões, serão alocadas v0 a v3 para as entradas e v4 a v7 para as saídas.



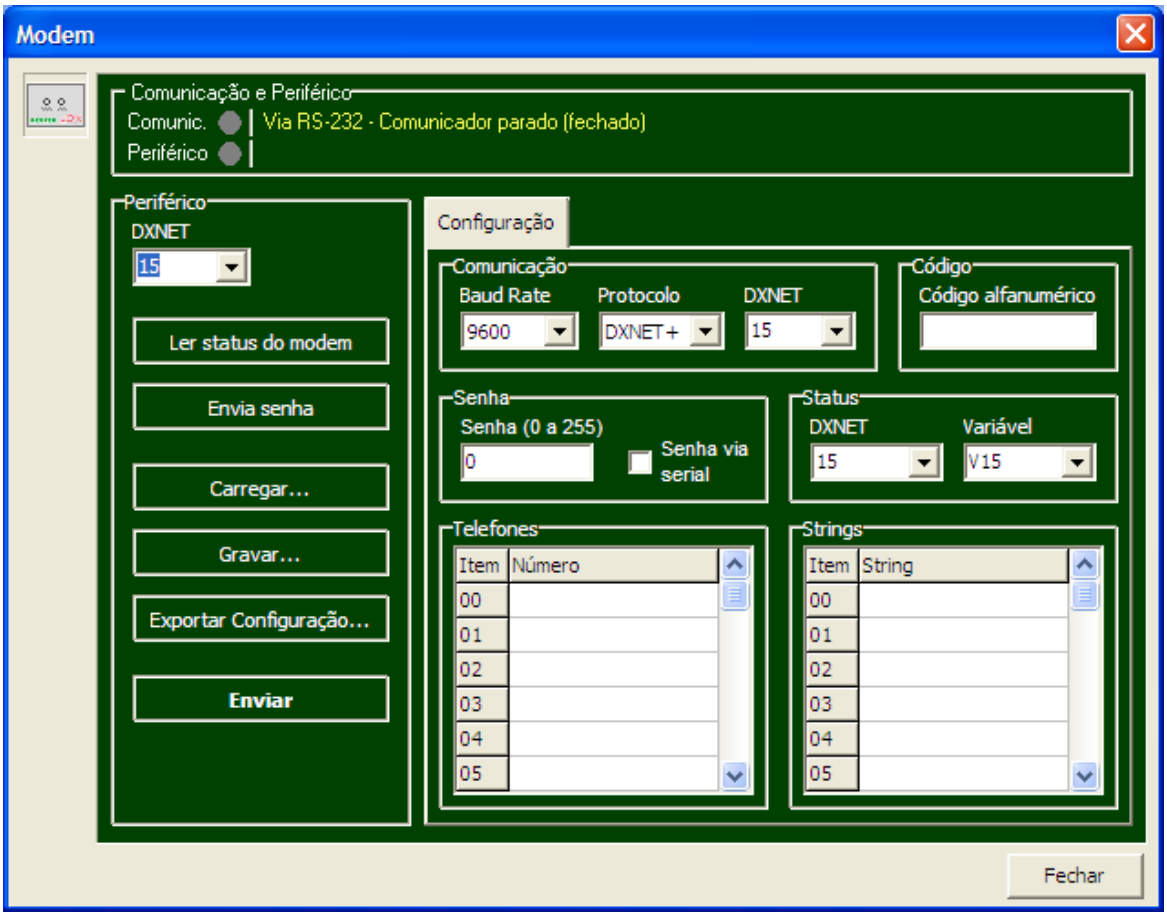
Periféricos

Esta opção acessa telas específicas para programação dos periféricos do controlador µDX100: Modem, IHM (Interface Homem/Máquina) e Conversor A/D (Analógico/Digital). Durante o uso destas telas o monitoramento do controlador é temporariamente interrompido, e por isso deve surgir uma janela com a seguinte advertência:

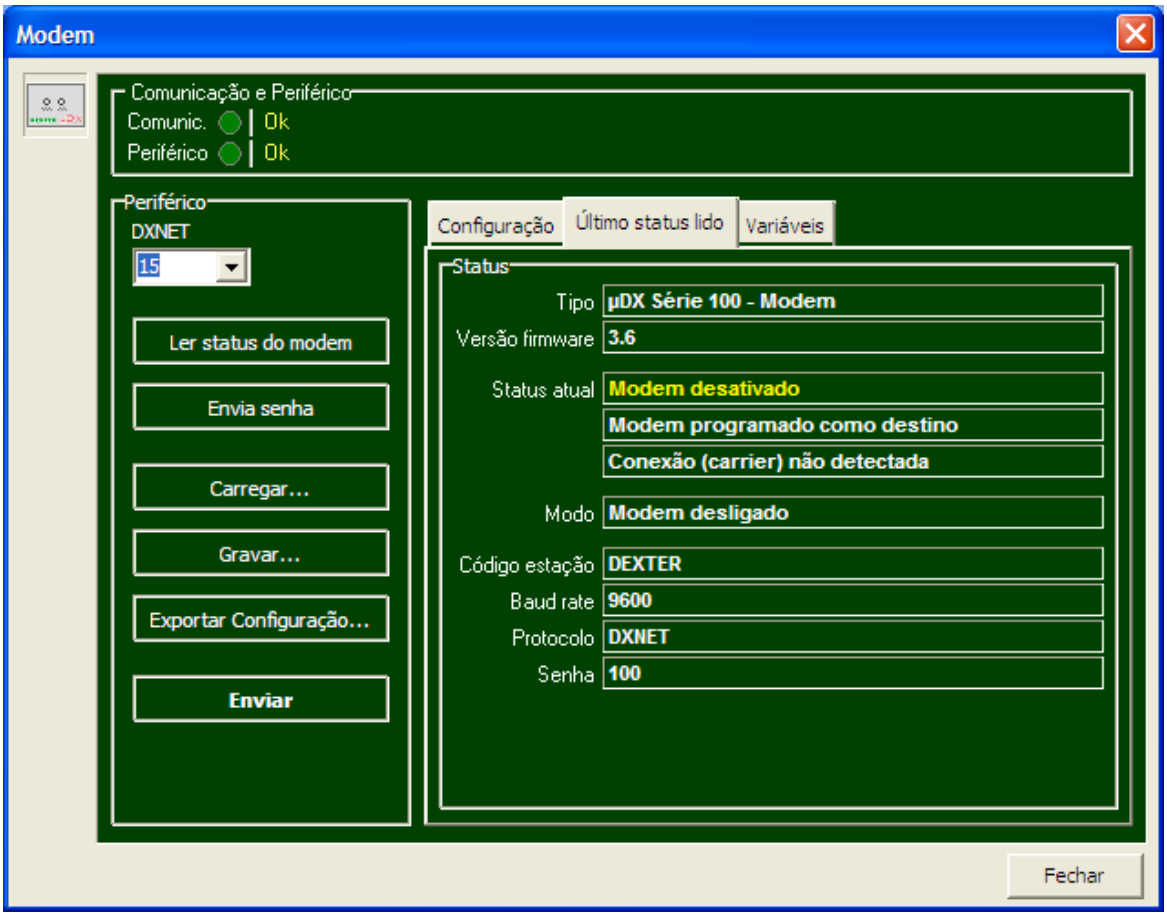


Modem

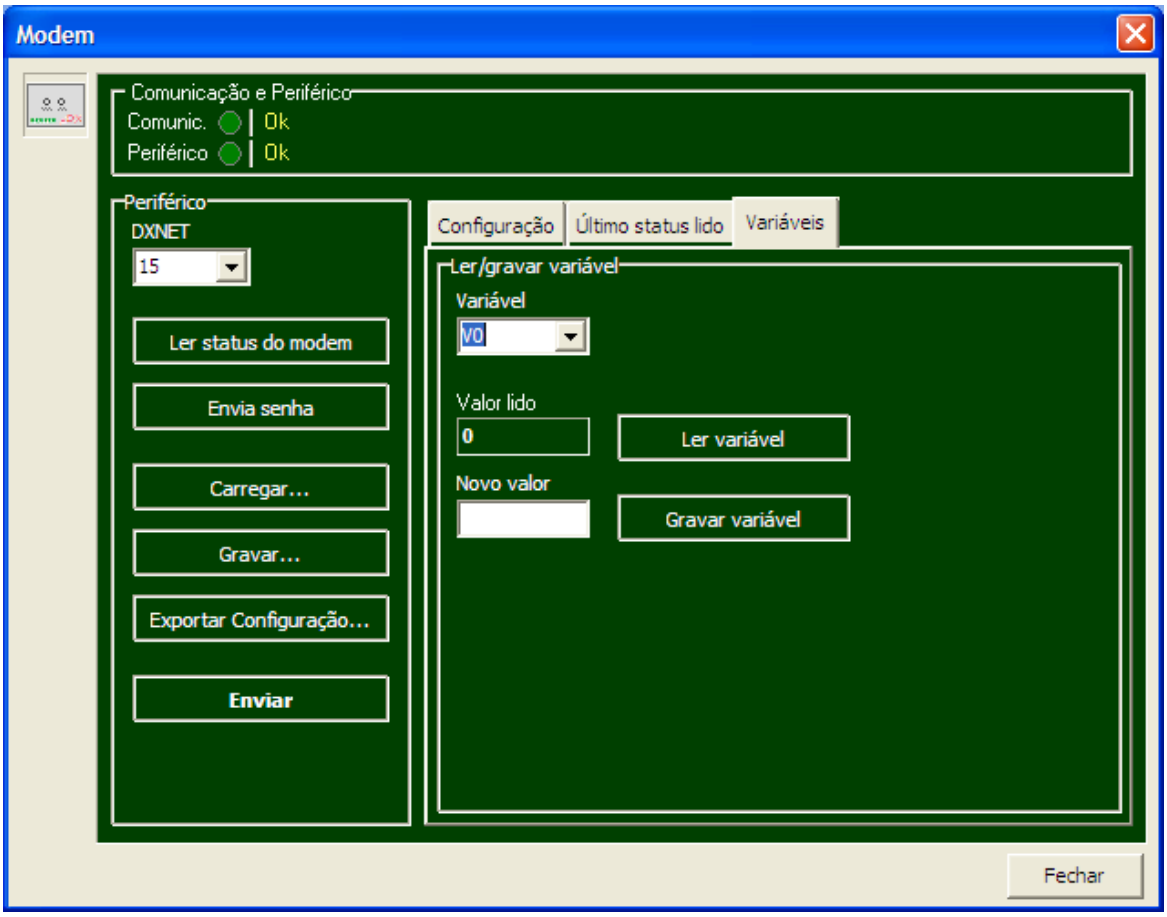
Ao selecionar a opção Modem surge a tela abaixo:



O primeiro item do quadro Periférico é o endereço DXNET que o PG irá usar para acessar o Modem. Todos os periféricos para µDX100 são fornecidos de fábrica com endereço DXNET 15. Certifique-se de que não existe outro periférico (Conversor A/D, IHM) ligado à rede DXNET e pressione a tecla **[Ler status do modem]**. Se a comunicação com o Modem for bem sucedida deverá surgir duas abas adicionais (**Último status lido** e **Variáveis**):



A aba **Último status lido**, como o nome já diz, traz informações sobre o status atual do Modem, como modo e baud rate (ver capítulo Periféricos → Modem). Já a aba **Variáveis** informa o valor das variáveis do Modem e também permite modificá-las:



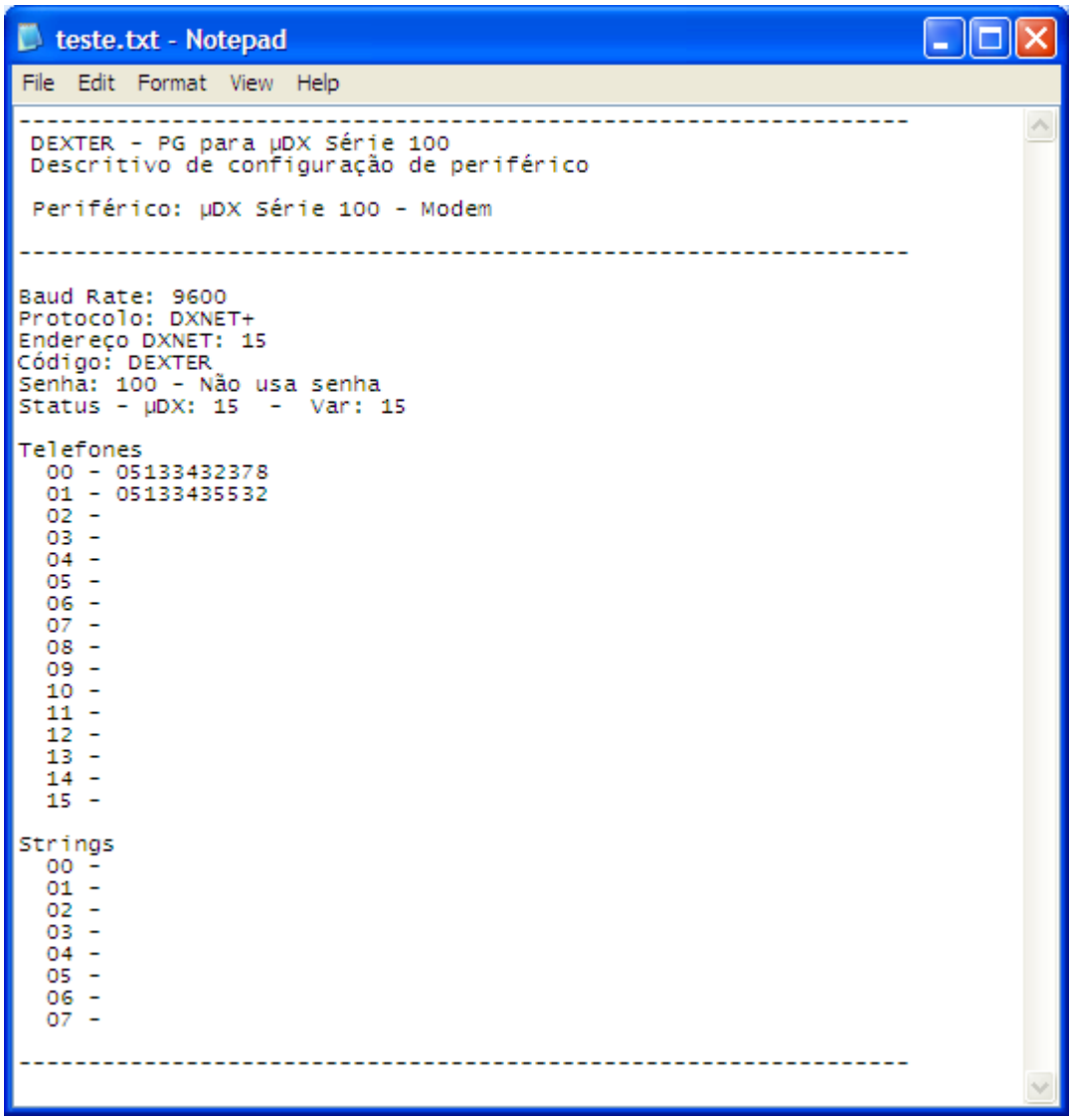
Note que a variável V0 do Modem determina seu modo de operação, conforme descrito no capítulo Periféricos → Modem.

Como é possível programar o Modem para que o mesmo exija senha para permitir comunicação o próximo item do quadro Periférico é a tecla **[Envia senha]**. Esta senha é especificada no menu Comunicação → Configurar Comunicador... → aba Modem. A tecla **[Envia senha]** transmite a senha especificada no comunicador para o Modem, permitindo o início de comunicação em Modems com a opção de senha via serial marcada.

Os botões **[Carregar...]** e **[Gravar...]** permitem ler ou salvar uma programação para Modem, respectivamente.

Atenção: sempre que a janela de Modem é fechada todos os dados configurados na mesma são perdidos. Então, caso se queira preservar estes dados é mandatório utilizar a opção **[Gravar...]**.

A opção **[Exportar Configuração...]** gera um arquivo de texto com a listagem de todas as configurações existentes na janela de Modem. Com isso é possível, por exemplo, imprimir estas configurações:



Por fim, o botão **[Enviar]** transmite os dados para o Modem. Note que o endereço acessado é determinado pelo endereço DXNET especificado no quadro **Periférico**. Já o endereço DXNET especificado na aba **Configuração** indica qual endereço o Modem deve assumir ao receber a configuração determinada nesta aba.

Atenção: Ao contrário do µDX100, o Modem não responde ao endereço 0 (zero) da DXNET, a menos que tenha sido programado para este endereço (o que não é aconselhável, pois irá conflitar com os controladores µDX100 na rede DXNET). O Modem é fornecido de fábrica no endereço DXNET 15.

Programação do Modem

A aba **Configuração** permite configurar o Modem, exceto quanto ao modo de operação, que é determinado pelo valor programado na variável v0 do Modem via aba **Variáveis**. Na aba **Configuração** temos os seguintes dados:

Baud Rate: especifica qual baud rate que o Modem irá assumir ao receber os dados (ao pressionar a tecla **[Enviar]**). Note que esta taxa de transmissão serve para conexão serial (via cabo RS232C ou RS485). A conexão via rede telefônica é sempre efetuada na taxa de 300 bps. Pode-se escolher as seguintes taxas de transmissão: 300, 600, 1200, 2400, 4800 e 9600 bps. Devido a limitações no processamento das comunicações no Modem é preciso usar dois stop bits

no caso de baud rate de 9600 bps. O comunicador do PG já seleciona automaticamente dois stop bits no caso desta taxa de comunicação.

Protocolo: permite escolher qual protocolo a ser usado com o Modem As opções suportadas são DXNET, DXNET+, DXNETX e MODBUS. Mas apenas os protocolos DXNET e DXNET+ são suportados pelo programa PG, de forma que se outro protocolo for especificado o PG irá perder comunicação com o Modem.

DXNET: indica qual o endereço que o Modem irá ocupar ao receber as configurações (ao pressionar a tecla **[Enviar]**). Esta tecla serve, portanto, para indicar ao modem que receber o programa qual o endereço na rede local DXNET ele deve assumir. Escolha um valor que não conflite com outros dispositivos ligados à rede DXNET.

Código alfanumérico: Além da senha do Modem existe ainda um código alfanumérico com 10 caracteres para segurança no acesso remoto. Ao receber uma chamada telefônica, o modem, após receber a chamada, aguarda pelo recebimento da senha e código do modem via telefone. Caso não receba estas informações ou as receba incorretas a ligação telefônica é derrubada. O código alfanumérico permite números, letras e caracteres especiais, como ponto, hífen, etc. Não é permitida acentuação no código. Com 10 caracteres a segurança é muito grande, pois o número de combinações é enorme.

Senha: Este campo permite programar uma senha de 8 bits (0 a 255) para o modem. Ao receber uma chamada telefônica, o modem, após atender a chamada, aguarda pelo recebimento da senha via telefone. Caso não a receba ou receba senha incorreta a ligação é derrubada. Existe a opção de marcar **[ ]Senha via serial**. Neste caso a senha é exigida mesmo em conexões seriais (RS232C ou RS485). Se o Modem for programado com esta opção ativada é preciso programar esta senha na **Configuração do Comunicador** do PG, e transmití-la ao iniciar comunicação via opção **Envia senha de comunicação** do menu pop-down **µDX**.

Status: os vários estados possíveis para o modem, assim como o estado de vários sinais desse, são disponíveis na variável v1 do modem. Esta variável pode ser replicada em uma variável de um µDX100 ligado à rede DXNET. O quadro **Status** permite especificar o endereço DXNET e o número da variável do µDX100 que irá receber o valor da variável v1 do Modem. Com isso, o µDX que irá controlar o modem tem acesso ao status dele. As informações contidas em v1 são as seguintes:

Variável v1	**bit 7**	Linha ocupada. Indica que a linha telefônica não está livre para utilização do modem.
	bit 6	Senha ok. Indica que a senha para modem (comunicação telefônica) foi recebida corretamente.
	bit 5	Discando. Modem está efetuando discagem.
	bit 4	Erro. Última comunicação transmitida não foi recebida corretamente.
	bit 3	Sinal de chamada (ring).
	bit 2	Portadora presente (carrier). O led vermelho no painel do modem também é ativado.
	bit 1	Origem. Indica que este modem está programado como origem da chamada telefônica (o outro modem deve estar programado como destino).
	bit 0	Modem ativado. O modem está ligado, e portanto ocupando a linha telefônica.

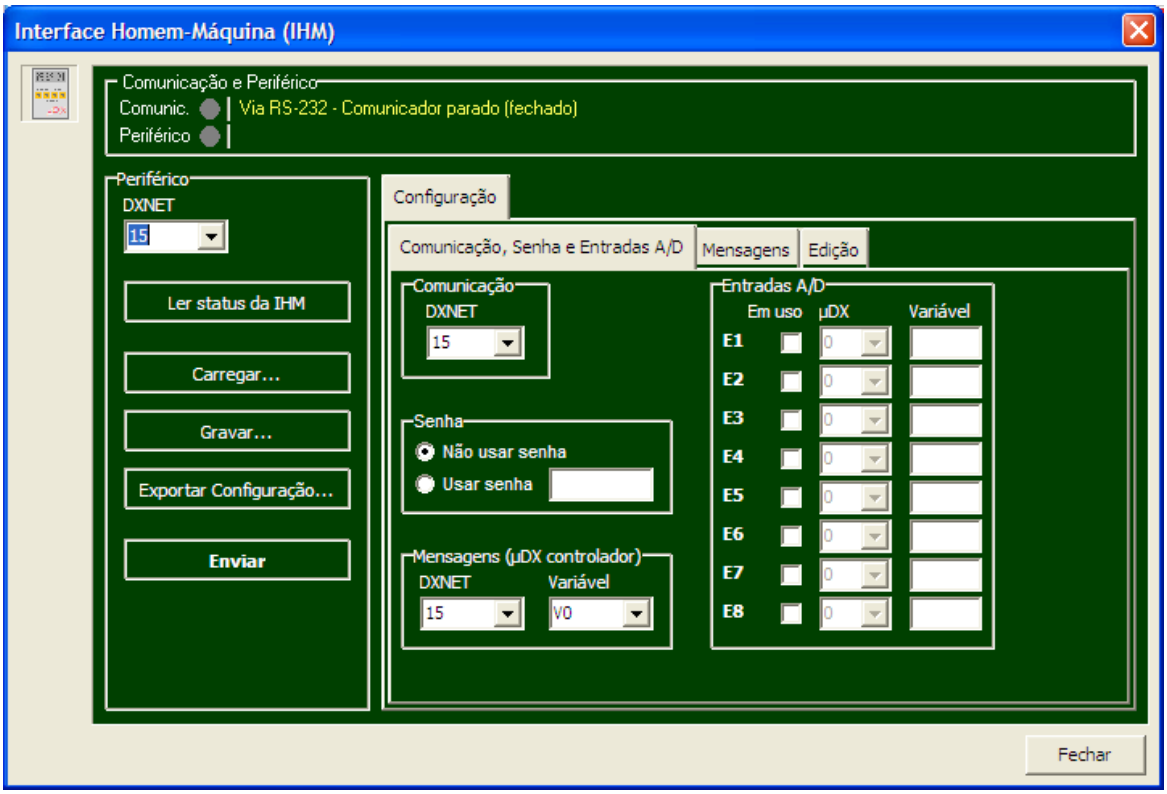
Telefones: Este campo permite programar até 16 números de telefone (cada um com até 16 dígitos) no modem, permitindo que um µDX100 ligado ao modem mande esse efetuar uma ligação telefônica para o número selecionado. No capítulo anterior Modem são descritos os vários valores da variável v0 do modem e como especificar qual número a ser usado na discagem. Utilize o mouse para selecionar um dos 16 números telefônicos possíveis. Para inserir uma pausa de 800ms entre um dígito e outro insira sinal de menos (-). Por exemplo, para ligar para a Dexter obtendo-se linha a partir de um ramal normalmente é preciso discar 0 (zero), aguardar a obtenção

de linha externa e então discar o número desejado. Para isso se especificaria o seguinte número: 0-33432378.

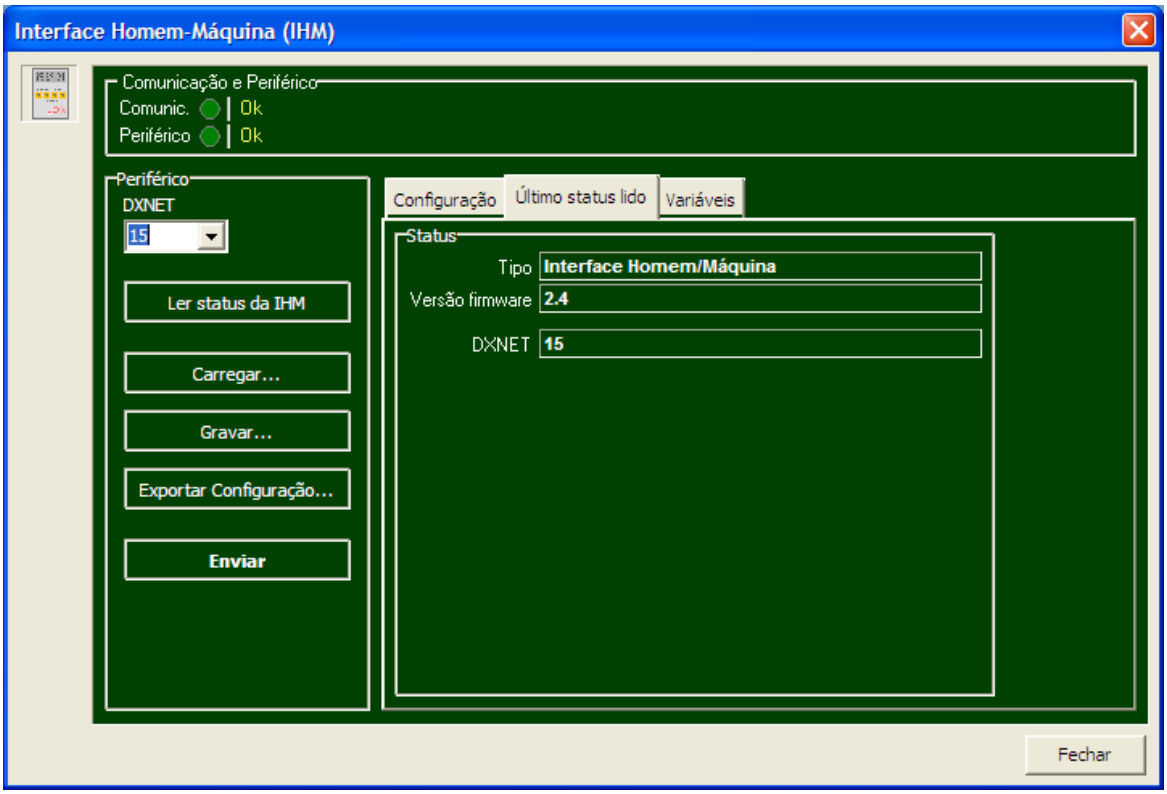
Strings: no caso de modems versão 3.4 ou superior é possível armazenar também 8 strings (cada um com 16 caracteres) para comandar modems externos via porta serial (via comandos AT). Isso é muito útil, por exemplo, para utilizar um telefone celular com modem interno para comunicação. Veja os estados 8 e 9 para maiores detalhes (no capítulo Modem).

IHM

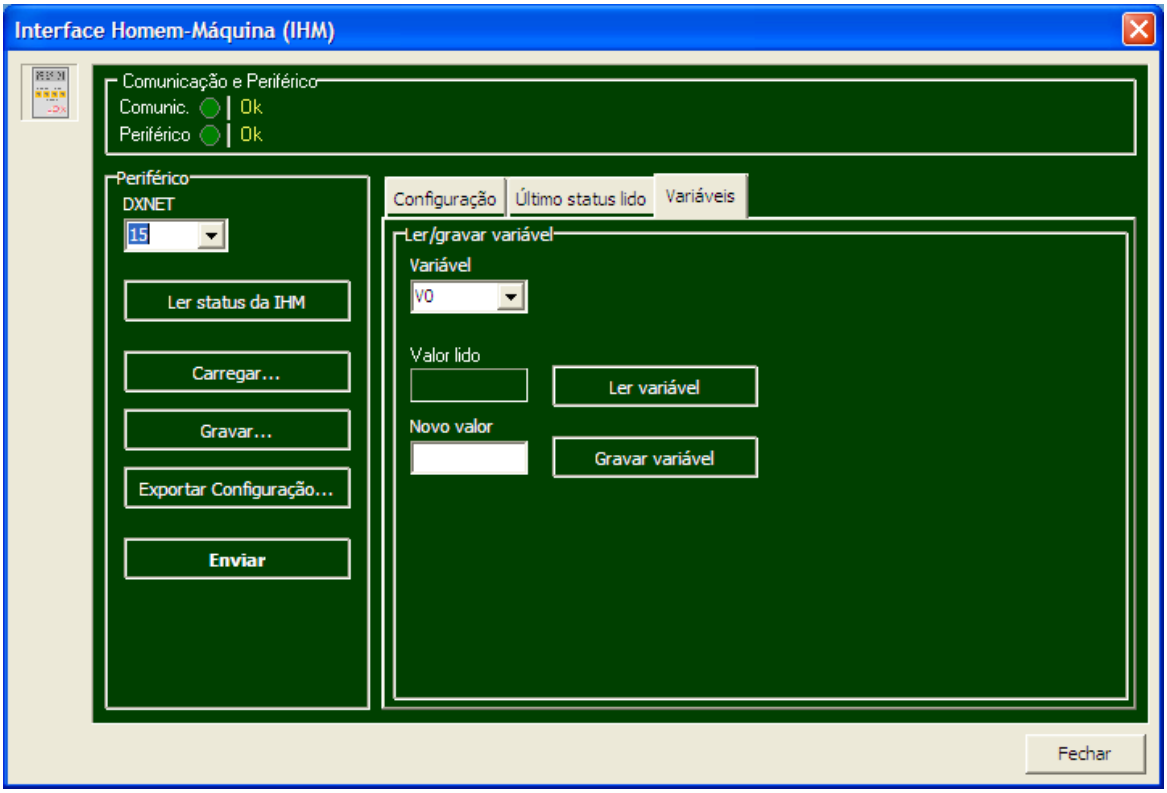
Ao selecionar a opção IHM surge a tela abaixo:



O primeiro item do quadro Periférico é o endereço DXNET que o PG irá usar para acessar a IHM (Interface Homem/Máquina). Todos os periféricos para µDX100 são fornecidos de fábrica com endereço DXNET 15. Certifique-se de que não existe outro periférico (Conversor A/D, Modem) ligado à rede DXNET e pressione a tecla **[Ler status da IHM]**. Se a comunicação com a IHM for bem sucedida deverá surgir duas abas adicionais (**Último status lido** e **Variáveis**):



A aba **Último status lido**, como o nome já diz, traz informações sobre o status atual da IHM, como versão de firmware e endereço DXNET (ver capítulo Periféricos → IHM). Já a aba **Variáveis** informa o valor das variáveis da IHM e também permite modificá-las:

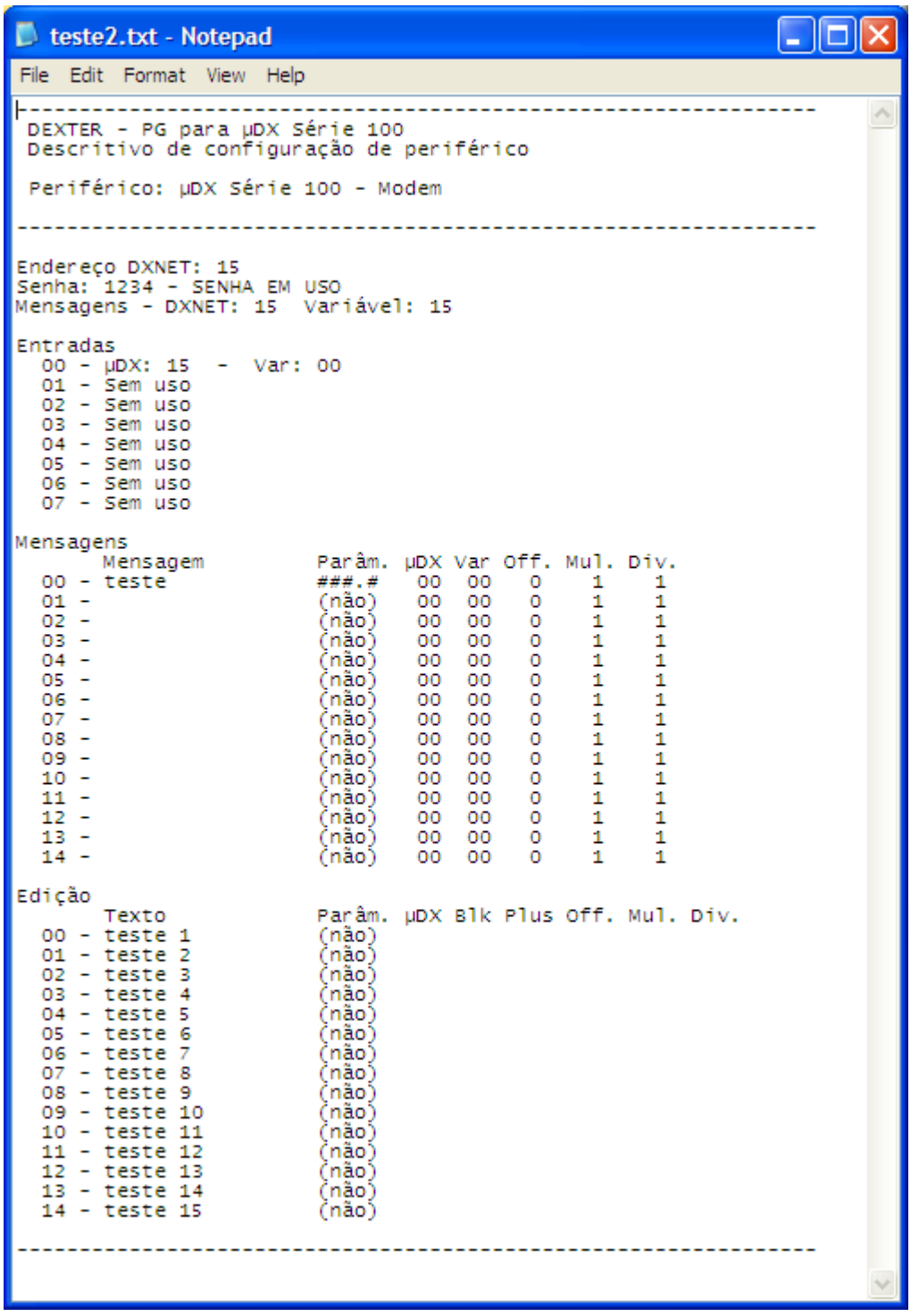


Note que a IHM possui 16 variáveis internas de 8 bits.

Os botões **[Carregar...]** e **[Gravar...]** permitem ler ou salvar uma programação para IHM, respectivamente.

Atenção: sempre que a janela de IHM é fechada todos os dados configurados na mesma são perdidos. Então, caso se queira preservar estes dados é mandatório utilizar a opção **[Gravar...]**.

A opção **[Exportar Configuração...]** gera um arquivo de texto com a listagem de todas as configurações existentes na janela de IHM. Com isso é possível, por exemplo, imprimir estas configurações:



teste2.txt - Notepad

File Edit Format View Help

```
------------------------------------------------------------------
 DEXTER - PG para µDX Série 100
 Descritivo de configuração de periférico

 Periférico: µDX Série 100 - Modem

------------------------------------------------------------------

Endereço DXNET: 15
Senha: 1234 - SENHA EM USO
Mensagens - DXNET: 15  Variável: 15

Entradas
  00 - µDX: 15  -  Var: 00
  01 - Sem uso
  02 - Sem uso
  03 - Sem uso
  04 - Sem uso
  05 - Sem uso
  06 - Sem uso
  07 - Sem uso

Mensagens
       Mensagem          Parâm. µDX Var Off. Mul. Div.
  00 - teste             ###.#   00  00   0    1    1
  01 -                   (não)   00  00   0    1    1
  02 -                   (não)   00  00   0    1    1
  03 -                   (não)   00  00   0    1    1
  04 -                   (não)   00  00   0    1    1
  05 -                   (não)   00  00   0    1    1
  06 -                   (não)   00  00   0    1    1
  07 -                   (não)   00  00   0    1    1
  08 -                   (não)   00  00   0    1    1
  09 -                   (não)   00  00   0    1    1
  10 -                   (não)   00  00   0    1    1
  11 -                   (não)   00  00   0    1    1
  12 -                   (não)   00  00   0    1    1
  13 -                   (não)   00  00   0    1    1
  14 -                   (não)   00  00   0    1    1

Edição
       Texto             Parâm. µDX Blk Plus Off. Mul. Div.
  00 - teste 1           (não)
  01 - teste 2           (não)
  02 - teste 3           (não)
  03 - teste 4           (não)
  04 - teste 5           (não)
  05 - teste 6           (não)
  06 - teste 7           (não)
  07 - teste 8           (não)
  08 - teste 9           (não)
  09 - teste 10          (não)
  10 - teste 11          (não)
  11 - teste 12          (não)
  12 - teste 13          (não)
  13 - teste 14          (não)
  14 - teste 15          (não)

------------------------------------------------------------------
```

Por fim, o botão **[Enviar]** transmite os dados para a IHM. Note que o endereço acessado é determinado pelo endereço DXNET especificado no quadro **Periférico**. Já o endereço DXNET especificado na aba **Configuração** indica qual endereço a IHM deve assumir ao receber a configuração determinada nesta aba.

Atenção: Ao contrário do µDX100, a IHM não responde ao endereço 0 (zero) da DXNET, a menos que tenha sido programado para este endereço (o que não é aconselhável, pois irá conflitar com os controladores µDX100 na rede DXNET). A IHM é fornecido de fábrica no endereço DXNET 15.

Programação da IHM

A aba **Configuração** permite configurar a IHM. Na aba **Configuração** temos três abas adicionais: **Comunicação, Senha e Entradas A/D**, **Mensagens** e **Edição**. Na primeira destas abas temos os seguintes itens:

DXNET: indica qual o endereço que a IHM irá ocupar ao receber as configurações (ao pressionar a tecla **[Enviar]**). Esta tecla serve, portanto, para indicar a IHM que receber o programa qual o endereço na rede local DXNET ele deve assumir. Escolha um valor que não conflite com outros dispositivos ligados à rede DXNET.

Senha: Este campo permite programar uma senha numérica de 4 dígitos (0 a 9999) para a IHM. Esta senha restringe a edição das constantes (set-points) do programa aplicativo do µDX100 via IHM. Caso a senha digitada não seja correta o operador pode visualizar estas constantes, mas é bloqueada a modificação de seus valores. A opção **Não usar senha** permite habilitar a edição de constantes sem a necessidade de senhas. No caso de programar-se senha na IHM, ao pressionar as teclas de Parâmetro da IHM, a senha é requerida. Note que as teclas de Parâmetro da IHM permitem escolher o valor de milhar e centena da senha, enquanto as teclas de Valor permitem escolher o valor de dezena e unidade da senha. Após especificar a senha no display da IHM aguarde alguns segundos para que esta avalie a senha escolhida. Se a senha escolhida for correta isso será declarado no display e a edição de constantes estará liberada. No caso de senha incorreta apenas a visualização do valor das constantes será permitido.

Mensagens: Esta tecla permite escolher qual variável de qual µDX100 irá especificar as mensagens a serem colocadas nas duas linhas do visor da IHM. As variáveis no µDX são de 8 bits. Assim, os 4 bits mais significativos especificam a mensagem a ser colocada na linha superior do display, enquanto os 4 bits menos significativos determinam a mensagem na linha inferior. Todos os 4 bits superiores ou inferiores ligados forçam linha superior ou inferior apagada, respectivamente. A seguir temos a tabela para os 4 bits superiores ou 4 bits inferiores da variável especificada via quadro **Mensagem** (endereço DXNET do µDX e variável deste µDX) e a mensagem que aparecerá no display da IHM.

Valor Binário	Valor Hexadecimal	Mensagem
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10

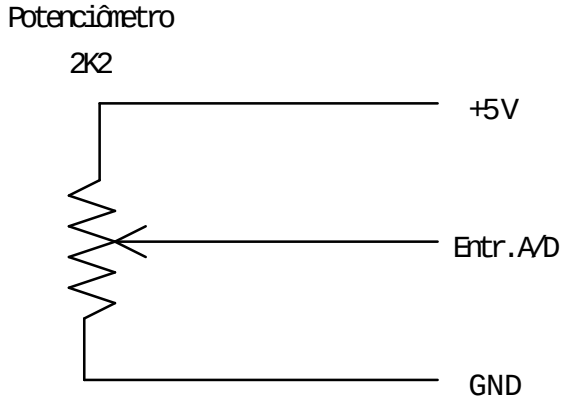
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	linha em branco

Por exemplo, se a variável especificada no µDX estiver com valor 243 = F3h = 11110011b a primeira linha do display permanecerá apagada, enquanto a segunda linha irá apresentar a mensagem 3.

Entradas A/D: Esta tecla permite especificar quais variáveis de quais µDXs ligados à rede DXNET irão receber os valores das conversões analógico/digitais efetuadas pela Interface Homem/Máquina. A IHM possui 8 entradas analógicas de 0 a 5 V (resolução de 8 bits = 19,6mV). Estas entradas podem ser ativadas ou inibidas individualmente.

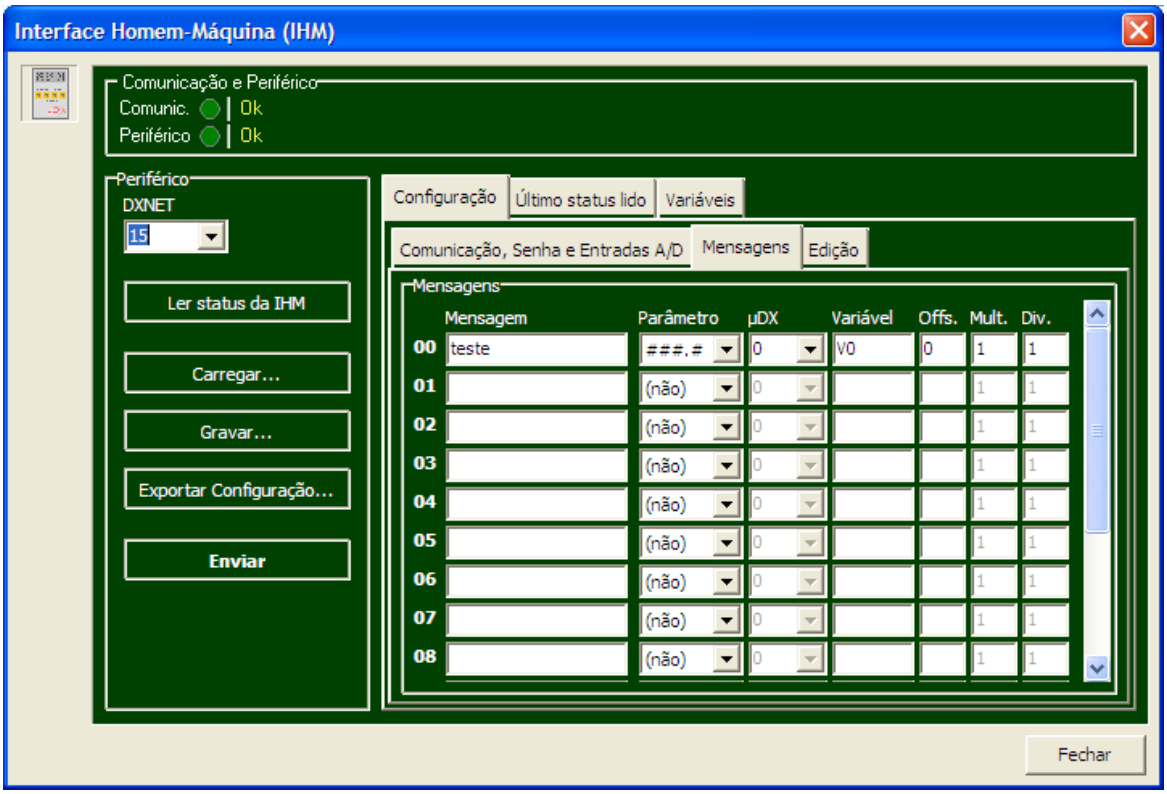
Já quanto a velocidade de conversão analógica/digital da IHM elas possuem taxas de conversão de 0,4 segundos por entrada analógica ativa. Assim, se houver apenas uma entrada analógica ativada o tempo de conversão será de apenas 0,4 segundos! No caso das 8 entradas analógicas ativadas, esse tempo aumenta para 8 x 0,4 s = 3,2 s.

Além das 8 entradas, o conector de entradas analógicas da IHM possui uma ligação à fonte de +5V e ao terra desta fonte. Isto permite conectar potenciômetros diretamente alimentados pela interface:



Note que cada entrada A/D possui um campo para endereço DXNET e um campo para especificar a variável que irá receber a conversão. Portanto, é possível transmitir as conversões para distintos controladores µDX100 existentes na rede DXNET, bastando especificar seus endereços de rede DXNET e variáveis usadas.

A aba seguinte é **Mensagens**, que contém todas as mensagens armazenadas na IHM:



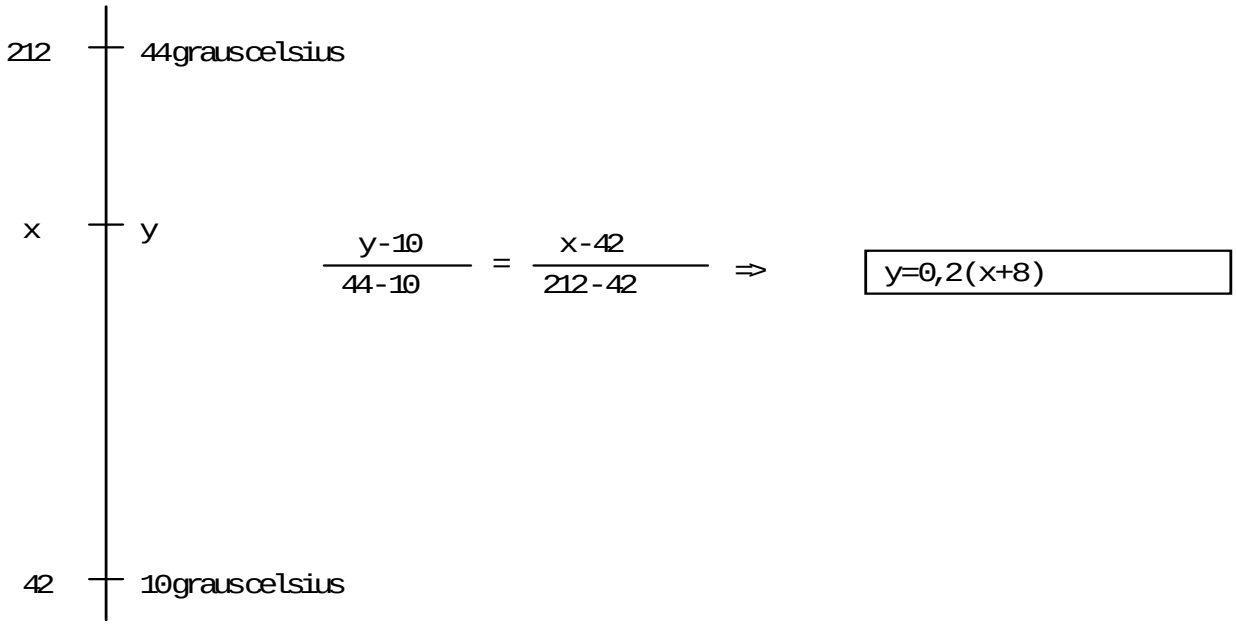
Esta aba permite programar até 15 mensagens (0 a 14) a serem apresentadas nas duas linhas do visor (display) alfanumérico da IHM. A mensagem para cada linha do display é determinada por uma variável do µDX ligado à IHM via DXNET. Esta variável é determinada no quadro **Mensagens** da aba anterior, **Comunicação, Senha e Entradas A/D**. Cada mensagem pode ter até 16 caracteres alfanuméricos (infelizmente, o display não permite acentuação e nem cedilha). Além disso, pode-se usar a mensagem para mostrar o valor de alguma variável de um µDX ligado à rede DXNET. O número da variável, assim como o endereço DXNET do µDX100, são especificados pelas colunas **µDX** e **Variável** da tabela. Este parâmetro adicionado à mensagem pode ser formatado conforme o ponto decimal requerido na coluna Parâmetro. Além disso, colunas adicionais permitem converter o valor lido na variável especificada em um valor já convertido na grandeza representada pela variável (por exemplo, temperatura). Vamos examinar com mais vagar estes recursos.

A coluna Parâmetro permite indicar se será usada uma variável de algum µDX para que seja visualizada junto com uma mensagem ou uma constante de algum µDX para edição na Interface Homem/Máquina (isso será visto quando examinarmos a próxima aba - **Edição**). Também possibilita fixar o ponto decimal. Após especificar qual o número da variável e o endereço DXNET do µDX100 que a possui, deve-se especificar o off-set e ganho para esta variável.

A posição do ponto decimal possui 5 possibilidades:

(não)	Sem variável monitorada na mensagem
###.#	Uma casa decimal
##.##	Duas casas decimais
#.###	Três casas decimais
.####	Quatro casas decimais
#####	Sem ponto decimal

O off-set e o ganho irão permitir transladar o valor da variável ou constante para o valor da correspondente grandeza física. Por exemplo, digamos que um PWM (veja manual do µDX) ligado a uma entrada do µDX converte a temperatura ambiente. Digamos que, para 10°C a variável associada ao bloco PWM assume valor 42, e para 44°C assume valor 212.



A equação de conversão do valor da variável lida pelo PWM para a temperatura correspondente é mostrada acima. Considerando que as variáveis ou constantes visualizadas no display da IHM têm ponto decimal fixo, com uma casa após a vírgula (###.#), a equação fica:

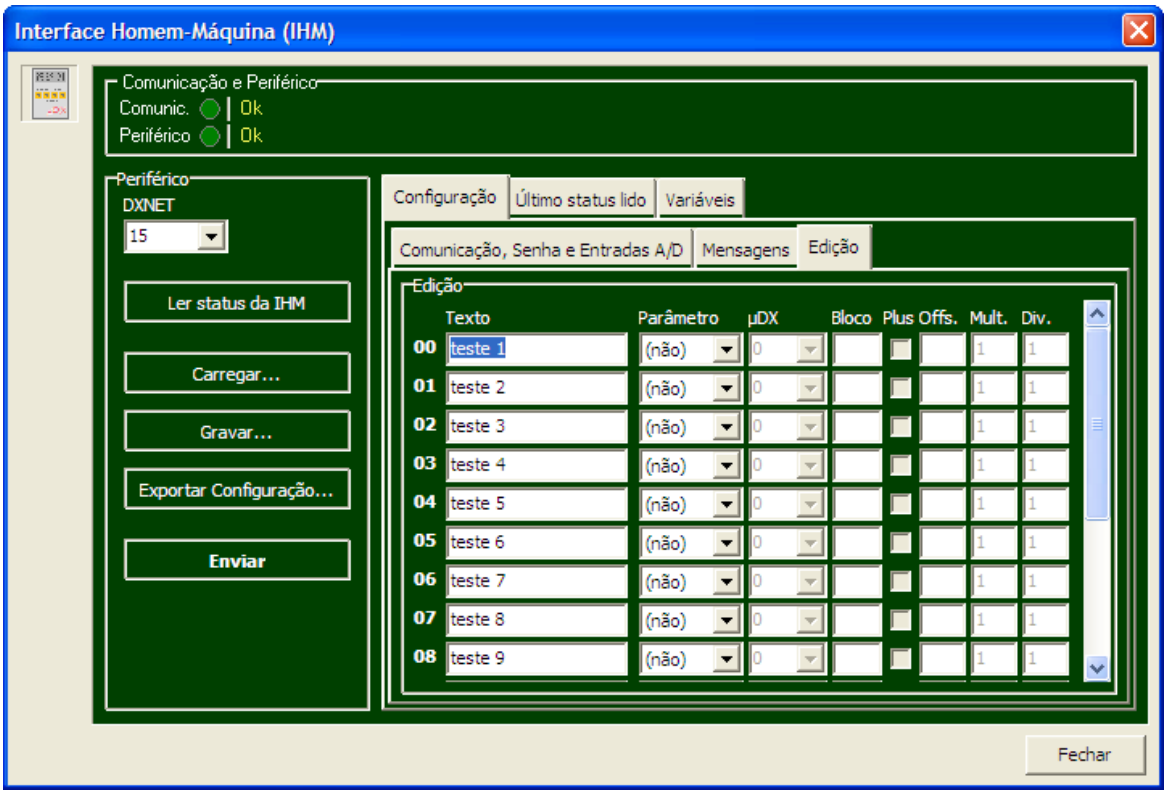
y = 2 (x + 8)

O valor 8 corresponde ao off-set, ou seja, o valor a ser somado à variável para sua conversão a unidade de temperatura. Já o valor 2 é o ganho, ou seja, o fator multiplicativo da conversão. Portanto, o display irá indicar o valor de temperatura ambiente de 10 a 44°C , com resolução de 0,2°C. Para isso basta especificar na programação da IHM off-set de 8 e ganho 2. Obviamente, neste caso o divisor seria 1.

Note que variáveis inseridas com mensagens sempre ocupam as 5 últimas posições da mensagem, reduzindo-a a 11 caracteres em vez de 16.

ATENÇÃO: A partir da versão 2.0.2.2 do PG para µDX100 existe um campo adicional nas Mensagens, designado como **16b** (16 bits). Caso esse campo seja marcado, a IHM irá ler variável word (16 bits), em vez de variável byte (8 bits). Tal leitura somente é permitida para IHM com versão igual ou superior a 2.6, e µDX101 versão 9.9 ou superior. O controlador µDX100 não suporta variáveis 16 bits.

Por fim, temos a aba **Edição**, que especifica constantes :(set-points) a serem visualizados e/ou editados na IHM:

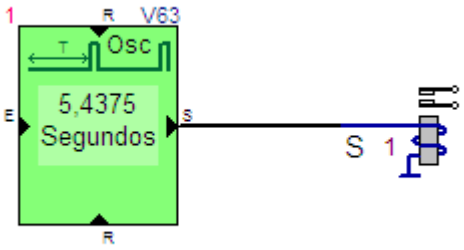


Além de 15 mensagens, a IHM também permite programar 16 mensagens de edição de constantes (0 a 15). A edição de constantes difere das mensagens em dois aspectos: em primeiro lugar, a edição só aparece no display da IHM quando são pressionadas as teclas de Parâmetro da interface (já as mensagens surgem no display conforme o valor de determinada variável de um µDX na rede DXNET). Além disso, na edição é possível modificar o valor de constantes existentes no programa dos controladores µDX ligados à IHM via DXNET (as mensagens permitem visualizar o valor de variáveis do µDX).

Note que uso a designação constante ou parâmetro do programa para µDX, indiferentemente, para especificar valores do programa gerado no PG, como tempo de blocos de Atraso, Mono, Oscilador ou Pulso, valor de comparação ou atribuição no bloco de Função, etc. Não confundir com as variáveis do µDX. As constantes do programa (16 constantes possíveis em um programa; de 0 a 15) são escolhidas especificando o número do bloco a ter sua constante editada. Este número sempre aparece no canto superior esquerdo dos blocos, após a pré-compilação.

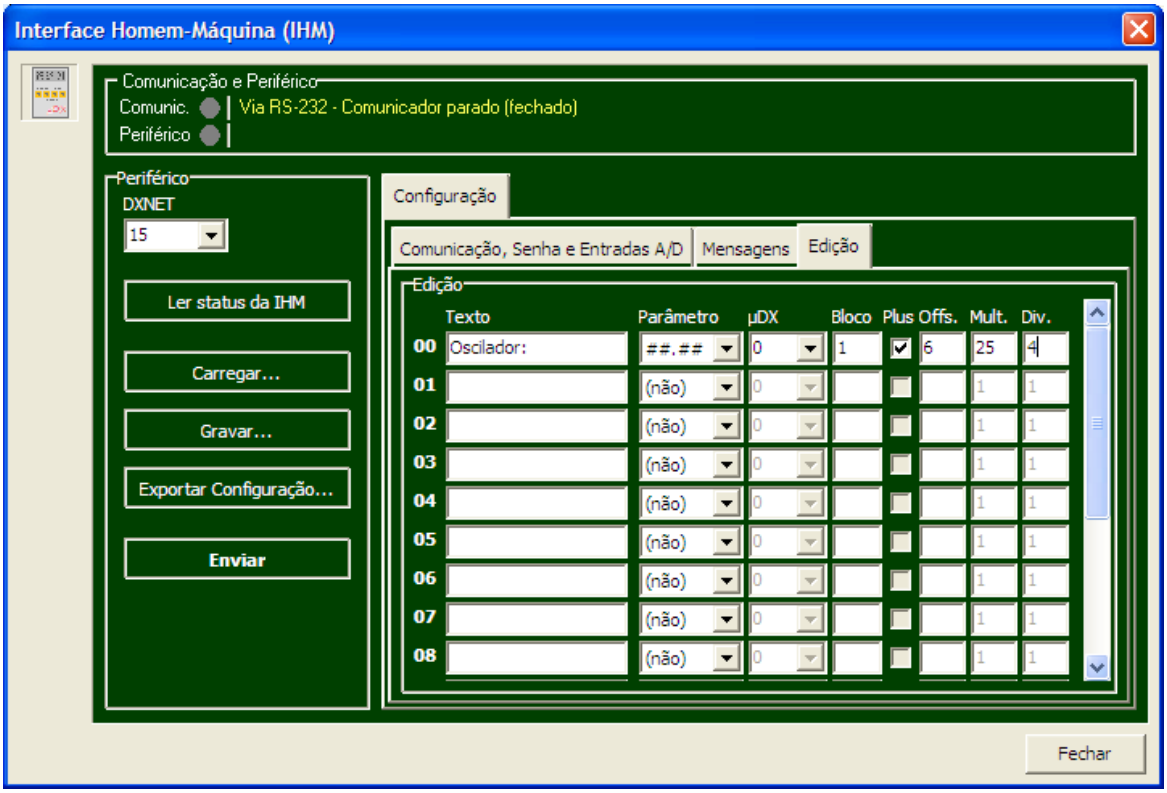
Importante: Sempre que o programa aplicativo do µDX for modificado é necessário verificar se a numeração dos blocos usados na edição via IHM não se modificaram, e proceder a correspondentes correções na coluna **Bloco** da aba **Edição** e retransmitir esta nova configuração para a IHM. Se isso não for feito a IHM irá acessar constantes em blocos quaisquer, que coincidentemente assumiram os números dos blocos originais.

ATENÇÃO: A partir da versão 2.1.0.0 do PG para µDX100 o campo **Plus** nas Edições foi substituído por **16b** (16 bits). Caso esse campo seja marcado, a IHM irá editar constante word (16 bits), em vez de constante byte (8 bits). Tal leitura somente é permitida para IHM com versão igual ou superior a 2.7, e µDX101 versão 9.9 ou superior. O controlador µDX100 não suporta variáveis 16 bits.



Exemplo de programa para µDX com número do bloco (bloco 1)

No exemplo acima temos um oscilador ligado diretamente a uma saída do µDX100. O valor de 5,4375s é a constante do oscilador, designada como constante associada ao bloco 1 (note que esta constante inicia com valor 87 : 1/16s x 87 = 5,4375s). Com a IHM, é possível modificar esta constante, mudando o período do oscilador.



Exemplo de edição de constante existente no bloco 1 do µDX100

Note que é possível converter um valor de constante para uma determinada unidade. No caso de utilizarmos o µDX100 na velocidade de 1/16s, o tempo do bloco de Atraso, na faixa de segundos, tem resolução de 0,0625s (1/16s). Colocando uma constante associada ao bloco de Atraso é possível modificar este tempo (de 0,0625s à 15,87s). Se desejarmos converter o valor da constante para o equivalente tempo, em centésimos de segundo, do bloco de atraso:

Tempo = 100 x 0,0625 x Constante

Tempo = 6,25 x Constante

Tempo = (625/100) x Constante

Tempo = (25/4) x Constante

Note que 625 é um número muito grande para o fator multiplicativo (este varia de 1 a 255). Mas

dividindo o numerador e o denominador pelo máximo divisor comum se obtem 25 para o fator multiplicativo, 4 para o fator de divisão, 0 para offset e duas casas decimais (##.##) para visualização da constante em segundos.

A seguir temos tabelas com o valor de Ponto Decimal, Ganho, Offset e Divisor para os diversos blocos temporizadores e ciclos de execução do Controlador µDX:

Ciclo de Execução de 1/16 s	
Temporizador	Programação da IHM
Monoestável, Atraso ou Pulso em segundos.	Ponto=##.##, Multiplicador=25, Offset=0, Divisor=4.
Monoestável, Atraso, Pulso ou Oscilador em minutos.	Ponto=##.##, Multiplicador=25, Offset=0, Divisor=1.
Monoestável ou Atraso em horas.	Ponto=##.##, Multiplicador=20, Offset=0, Divisor=3.
Oscilador em segundos.	Ponto=##.##, Multiplicador=25, Offset=6, Divisor=4.

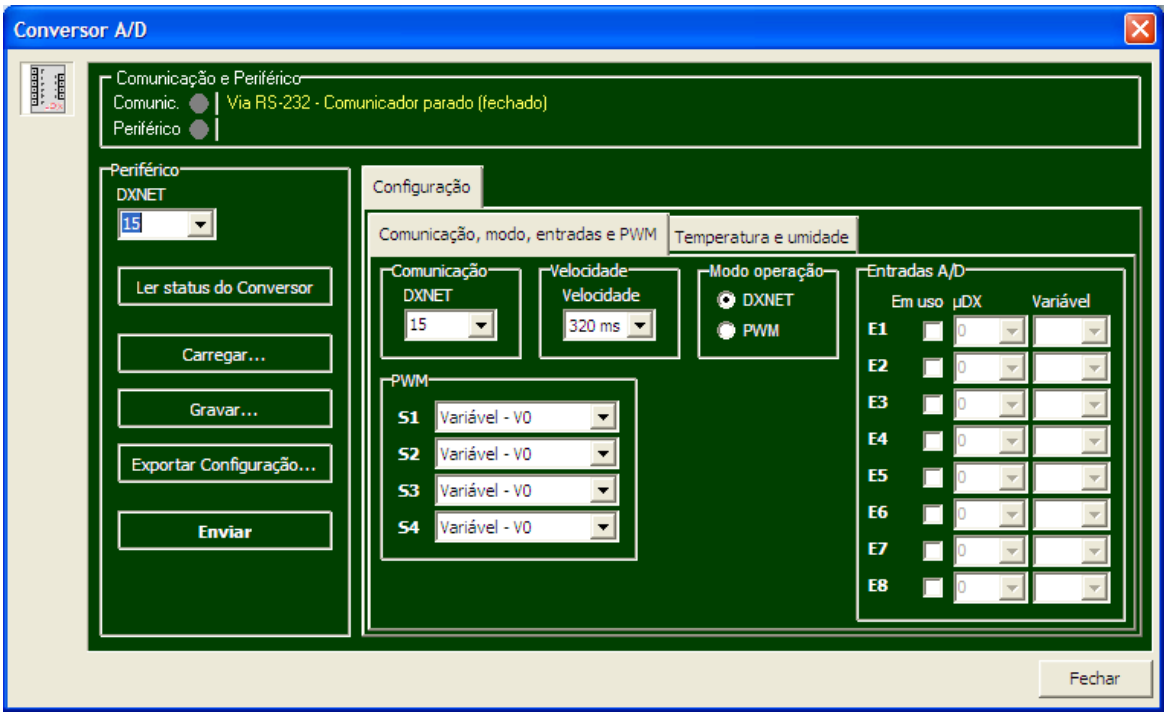
Ciclo de Execução de 1/32 s	
Temporizador	Programação da IHM
Monoestável, Atraso ou Pulso em segundos.	Ponto=#.###, Multiplicador=125, Offset=0, Divisor=4.
Monoestável, Atraso, Pulso ou Oscilador em minutos.	Ponto=##.##, Multiplicador=25, Offset=0, Divisor=2.
Monoestável ou Atraso em horas.	Ponto=#.###, Multiplicador=100, Offset=0, Divisor=3.
Oscilador em segundos.	Ponto=#.###, Multiplicador=125, Offset=31, Divisor=4.

Ciclo de Execução de 1/64 s	
Temporizador	Programação da IHM
Monoestável, Atraso ou Pulso em segundos.	Ponto=#.###, Multiplicador=125, Offset=0, Divisor=8.
Monoestável, Atraso, Pulso ou Oscilador em minutos.	Ponto=##.##, Multiplicador=25, Offset=0, Divisor=4.
Monoestável ou Atraso em horas.	Ponto=#.###, Multiplicador=50, Offset=0, Divisor=3.
Oscilador em segundos.	Ponto=#.###, Multiplicador=125, Offset=16, Divisor=8.

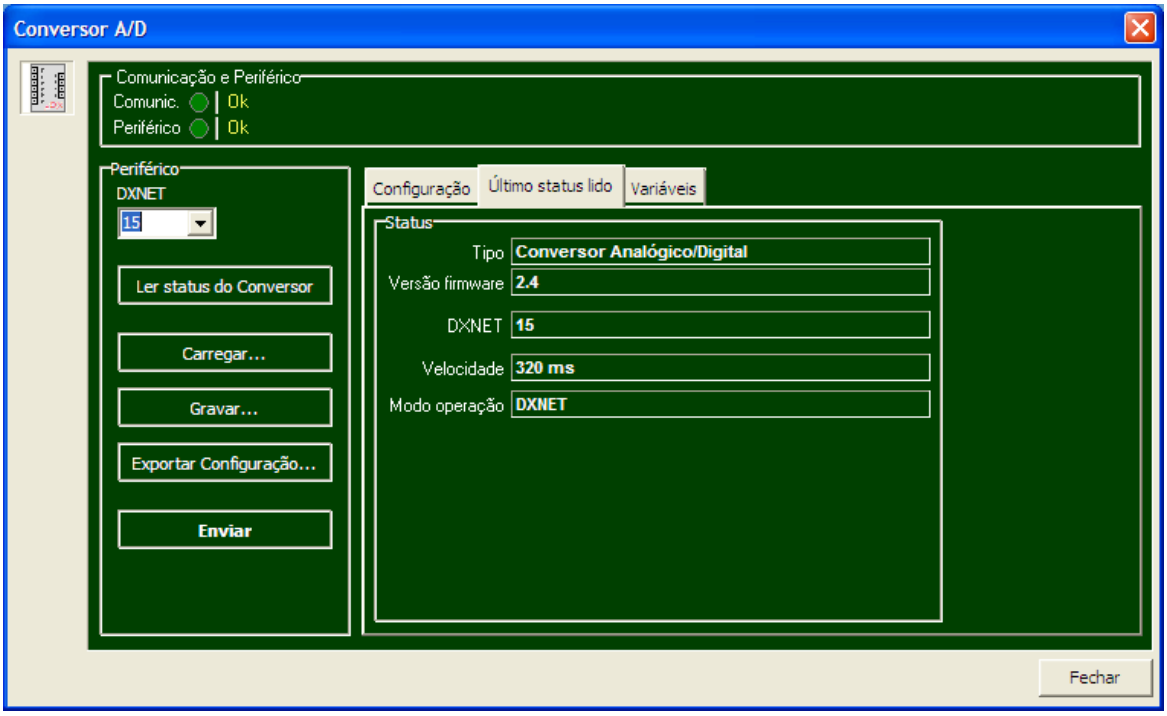
Ciclo de Execução de 1/256 s	
Temporizador	Programação da IHM
Monoestável, Atraso ou Pulso em segundos.	Ponto=#.###, Multiplicador=125, Offset=0, Divisor=32.
Monoestável, Atraso, Pulso ou Oscilador em minutos.	Ponto=#.###, Multiplicador=125, Offset=0, Divisor=8.
Monoestável ou Atraso em horas.	Ponto=#.###, Multiplicador=25, Offset=0, Divisor=6.
Oscilador em segundos.	Ponto=#.###, Multiplicador=125, Offset=4, Divisor=32.

Conversor A/D

Ao selecionar a opção Conversor A/D surge a tela abaixo:

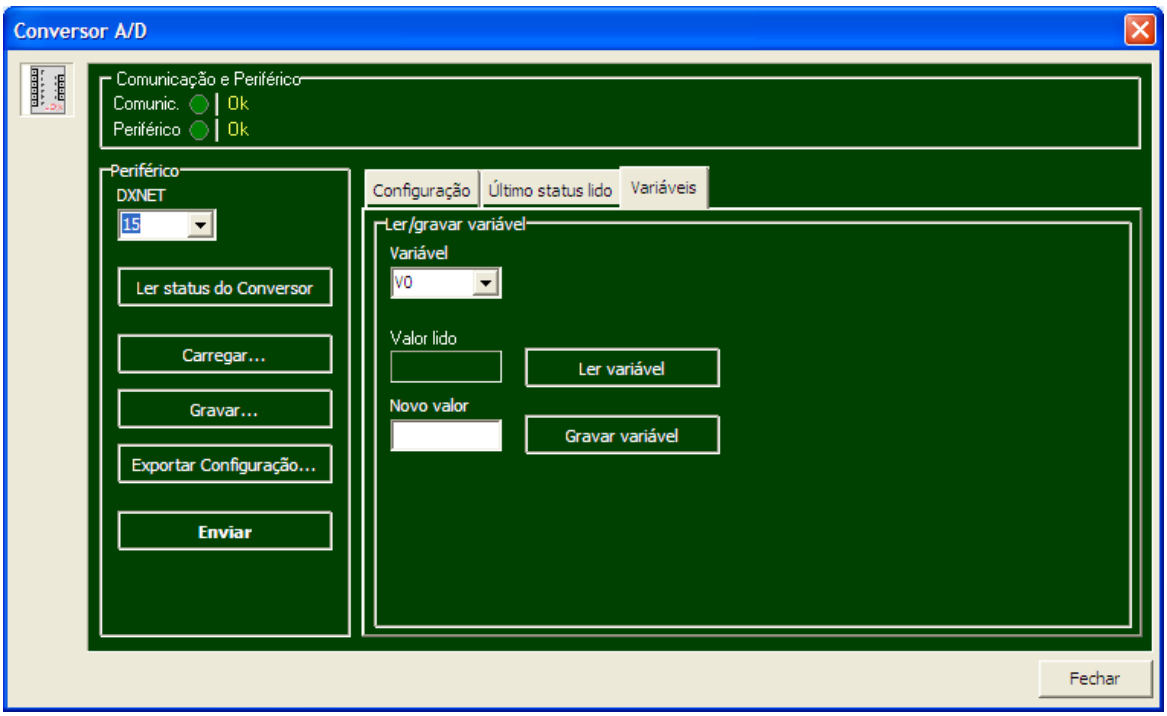


O primeiro item do quadro Periférico é o endereço DXNET que o PG irá usar para acessar o Conversor A/D (Conversor Analógico/Digital). Todos os periféricos para µDX100 são fornecidos de fábrica com endereço DXNET 15. Certifique-se de que não existe outro periférico (IHM, Modem) ligado à rede DXNET e pressione a tecla **[Ler status do Conversor]**. Se a comunicação com o Conversor A/D for bem sucedida deverá surgir duas abas adicionais (**Último status lido** e **Variáveis**):



A aba **Último status lido**, como o nome já diz, traz informações sobre o status atual do

Conversor A/D, como versão de firmware e endereço DXNET (ver capítulo Periféricos → Conversor A/D). Já a aba **Variáveis** informa o valor das variáveis do Conversor A/D e também permite modificá-las:

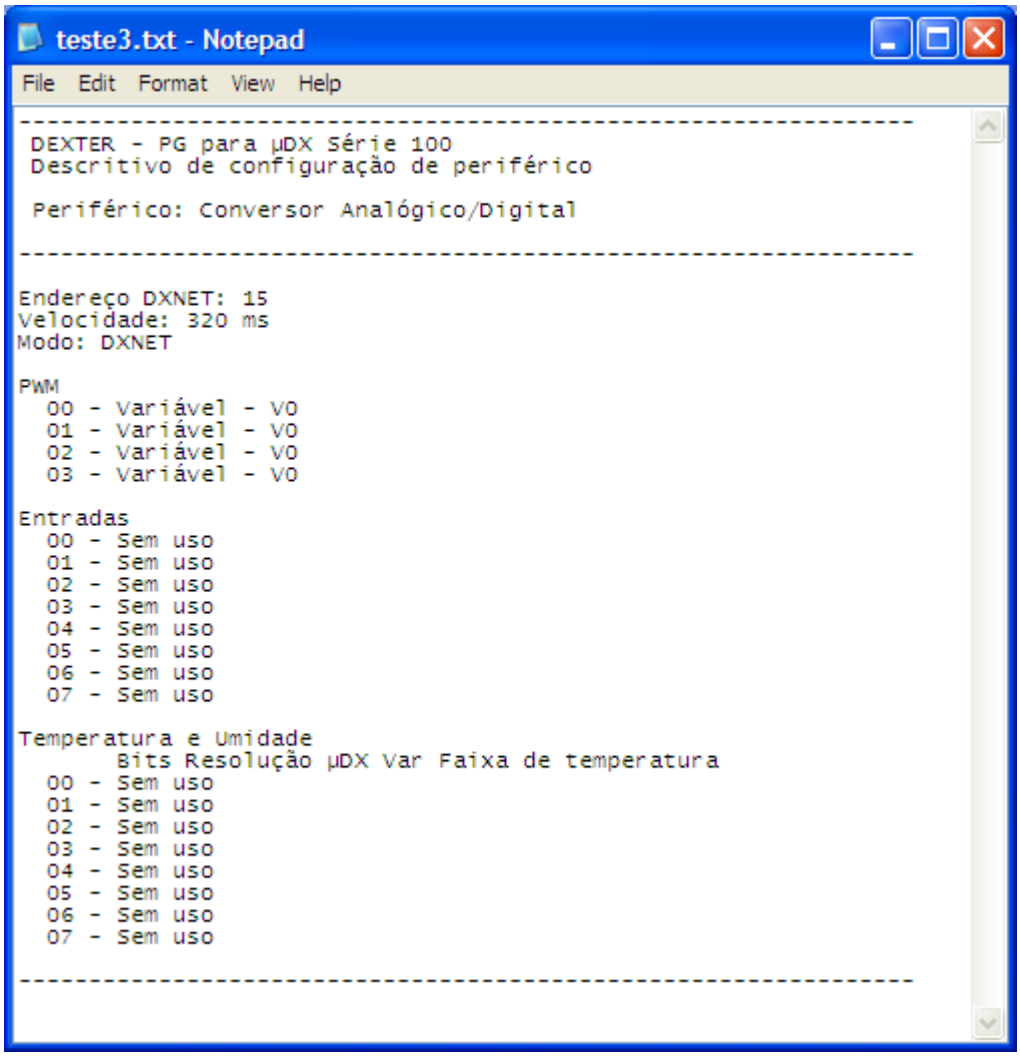


Note que o Conversor A/D possui 16 variáveis internas de 8 bits.

Os botões **[Carregar...]** e **[Gravar...]** permitem ler ou salvar uma programação para Conversor A/D, respectivamente.

Atenção: sempre que a janela de Conversor A/D é fechada todos os dados configurados na mesma são perdidos. Então, caso se queira preservar estes dados é mandatório utilizar a opção **[Gravar...]**.

A opção **[Exportar Configuração...]** gera um arquivo de texto com a listagem de todas as configurações existentes na janela de IHM. Com isso é possível, por exemplo, imprimir estas configurações:



Por fim, o botão **[Enviar]** transmite os dados para o Conversor A/D. Note que o endereço acessado é determinado pelo endereço DXNET especificado no quadro **Periférico**. Já o endereço DXNET especificado na aba **Configuração** indica qual endereço o Conversor A/D deve assumir ao receber a configuração determinada nesta aba.

Atenção: Ao contrário do µDX100, o Conversor A/D não responde ao endereço 0 (zero) da DXNET, a menos que tenha sido programado para este endereço (o que não é aconselhável, pois irá conflitar com os controladores µDX100 na rede DXNET). O Conversor A/D é fornecido de fábrica no endereço DXNET 15.

Programação do Conversor A/D

A aba **Configuração** permite configurar o Conversor A/D. Na aba **Configuração** temos duas abas adicionais: **Comunicação, modo, entradas e PWM**, **Temperatura e umidade**. Na primeira destas abas temos os seguintes itens:

DXNET: indica qual o endereço que o Conversor A/D irá ocupar ao receber as configurações (ao pressionar a tecla **[Enviar]**). Esta tecla serve, portanto, para indicar o Conversor A/D que receber o programa qual o endereço na rede local DXNET ele deve assumir. Escolha um valor que não conflite com outros dispositivos ligados à rede DXNET.

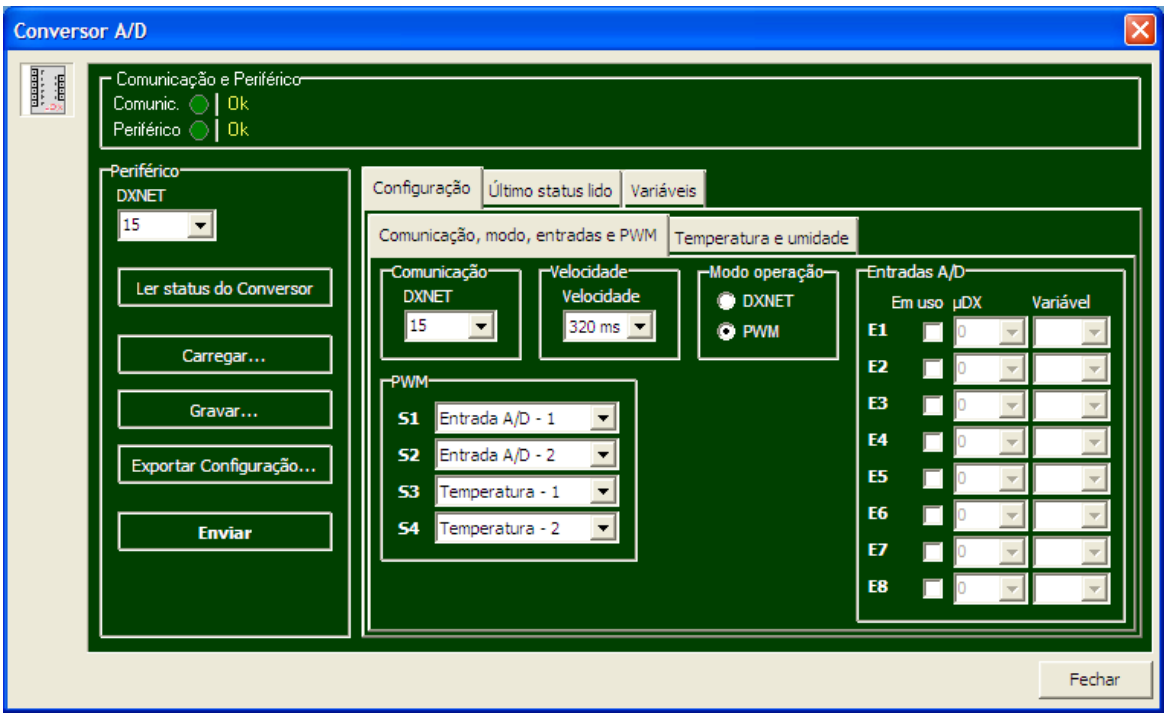
Velocidade: Os Conversores A/D atuais (versão de firmware 1.9 ou posterior) permitem modular a velocidade de conversão das entradas analógicas. Assim, é possível selecionar os seguintes períodos de conversão: 320ms, 160ms, 80ms e 40ms. A medida que diminuímos o período o

Conversor A/D torna-se mais rápido, mas em contrapartida aumentamos o uso da rede DXNET, e com isso aumenta a possibilidade de conflitos na comunicação (principalmente se a rede DXNET tiver muitos equipamentos se comunicando). Note que o tempo de varredura total das entradas analógicas é obtido multiplicando-se a velocidade programada (40 a 320ms) pela quantidade de entradas analógicas habilitadas. Por exemplo, no caso das 8 entradas analógicas habilitadas e velocidade de conversão de 320ms, a varredura total levará 320ms x 8 = 2,56s. Já no caso de apenas uma entrada analógica habiltada, e velocidade de conversão de 40ms, o tempo de varredura será de apenas 40ms. No caso de uso de sensores de umidade a velocidade é fixada compulsoriamente em 320ms.

Modo operação: O Conversor Analógico/Digital pode transmitir os dados para controladores programáveis µDX100 por duas formas: ou via rede DXNET ou via modulação por largura de pulso (PWM). Via rede DXNET o conversor utiliza um endereço na rede DXNET e transmite para as variáveis dos µDXs os valores de temperatura e entradas analógicas. Já no caso de modulação por largura de pulso (PWM), é necessário ligar as 4 saídas PWM disponíveis no conversor as entradas dos µDXs. No programa do µDX usa-se o bloco PWM para converter os pulsos em um valor de 0 a 255. Note que este método restringe o Conversor A/D a apenas 4 variáveis analógicas (entradas analógicas ou temperaturas), já que só existem 4 saídas PWM. Portanto, a modulação por largura de pulso só deve ser utilizada quando não houver endereço disponível para o Conversor A/D na rede DXNET. Outra limitação é que esta modulação só funciona para µDXs de versão igual ou superior que 5.7. O programa PG inicializa sempre como transmissão DXNET. Note que os dois modos de transmissão são mutuamente excludentes, ou seja, se for selecionado modo DXNET as saídas PWM não funcionam e vice-versa.

Atualmente, a entrada de sinais analógicos via modulação por largura de pulso no Controlador µDX está obsoleta, pois a transmissão via rede DXNET é mais precisa e confiável, e os Conversores A/D atuais permitem modular a velocidade de conversão (vide tecla [Veloc.]). Assim, normalmente, é utilizado o modo de transmissão via rede DXNET. Neste caso, as saídas PWM podem ser associadas a variáveis do Conversor A/D e, com uso da Placa de Saída Digital/Analógica, obter-se um sinal analógico de saída de 0 a 10V, ou de 0 a 5V.

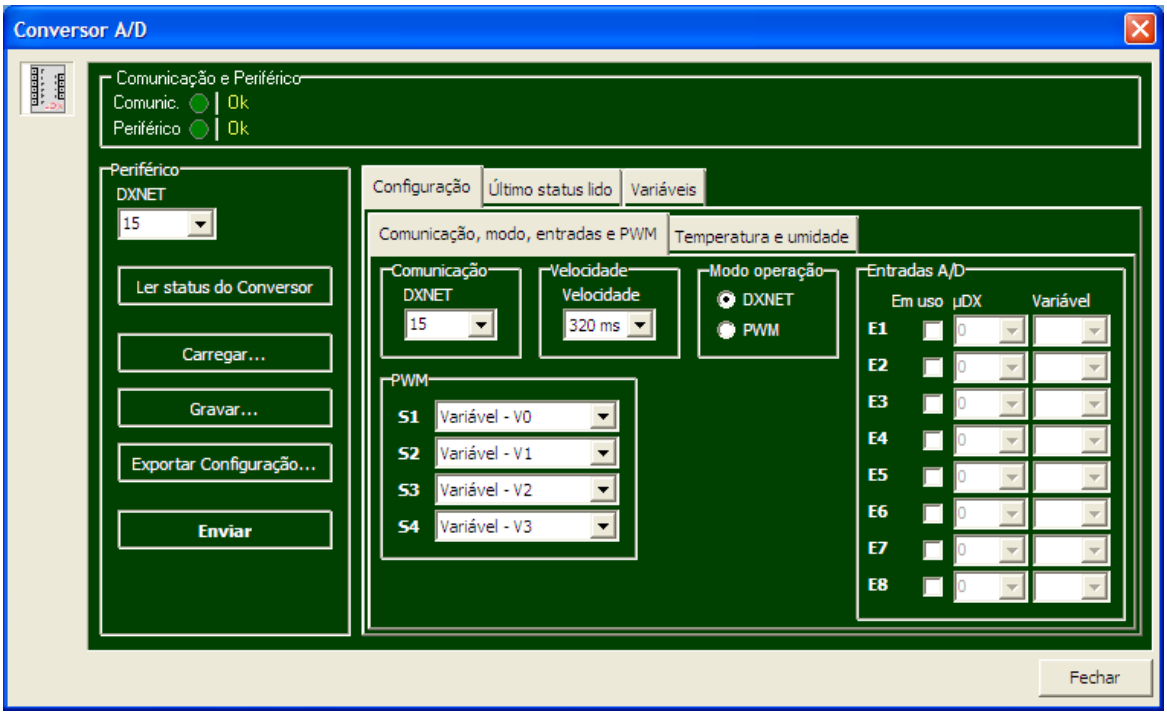
Caso seja especificada transmissão via modulação por largura de pulso (PWM) é necessário especificar que valor será atribuído para cada uma das 4 saídas PWM. Então, é possível escolher entre as 8 entradas analógicas e os 8 sensores de temperatura:



Note que no caso de modulação por largura de pulso (PWM) só é possível transmitir temperaturas em 8 bits. Caso seja selecionado resolução de 16 bits para o sensor de temperatura

apenas o byte mais significativo (MSB) será considerado.

No caso, mais usual, de modo de transmissão via rede DXNET, a tecla [PWM] permite selecionar 4 das 16 variáveis internas do Conversor A/D (v0 a v15), de forma que o valor destas 4 variáveis determinem a modulação por largura de pulso presente em cada saída PWM. Se for ligada uma Placa de Saída Digital/Analógica a cada saída PWM do Conversor A/D, obtem-se um sinal analógico de 0-10V ou 0-5V, proporcional ao valor da variável associada (ver capítulo Conversor A/D). Ou seja, o Controlador µDX pode comandar saídas analógicas simplesmente escrevendo novos valores nas variáveis do Conversor A/D:



Entradas A/D: Esta tecla permite especificar quais variáveis de quais µDXs ligados à rede DXNET irão receber os valores das conversões analógico/digitais efetuadas pela Conversor A/D. O Conversor A/D possui 8 entradas analógicas de 0 a 5 V (resolução de 8 bits = 19,6mV), 0 a 10V (resolução de 39mV), ou ainda 0 a 20mA (resolução de 78µA). Estas entradas podem ser ativadas ou inibidas individualmente. Note que cada entrada A/D possui um campo para endereço DXNET e um campo para especificar a variável que irá receber a conversão. Portanto, é possível transmitir as conversões para distintos controladores µDX100 existentes na rede DXNET, bastando especificar seus endereços de rede DXNET e variáveis usadas.

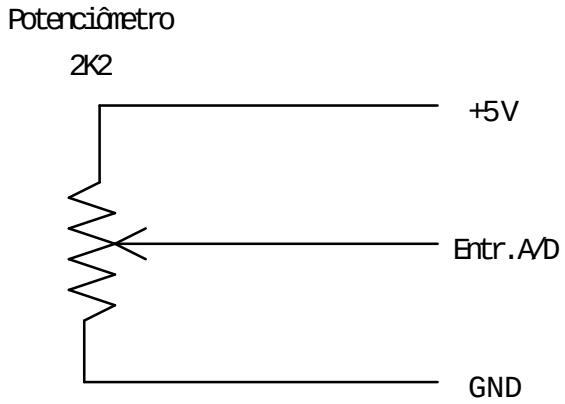
Esta tecla permite ativar as entradas analógicas do conversor A/D e especificar quais variáveis de quais µDXs ligados à rede DXNET irão receber os valores das conversões analógico/digitais efetuadas. A conversão é em 8 bits. O conversor A/D possui 8 entradas analógicas programáveis via estrapes (jumpers) na placa impressa para 3 níveis de entrada de sinal:

0 a 5 Vdc	→	resolução de 5/255 = 19,608 mV
0 a 10 Vdc	→	resolução de 10/255 = 39,216 mV
0 a 20 mA	→	resolução de 20/255 = 78,431 µA

As resistências de entrada para cada nível são:

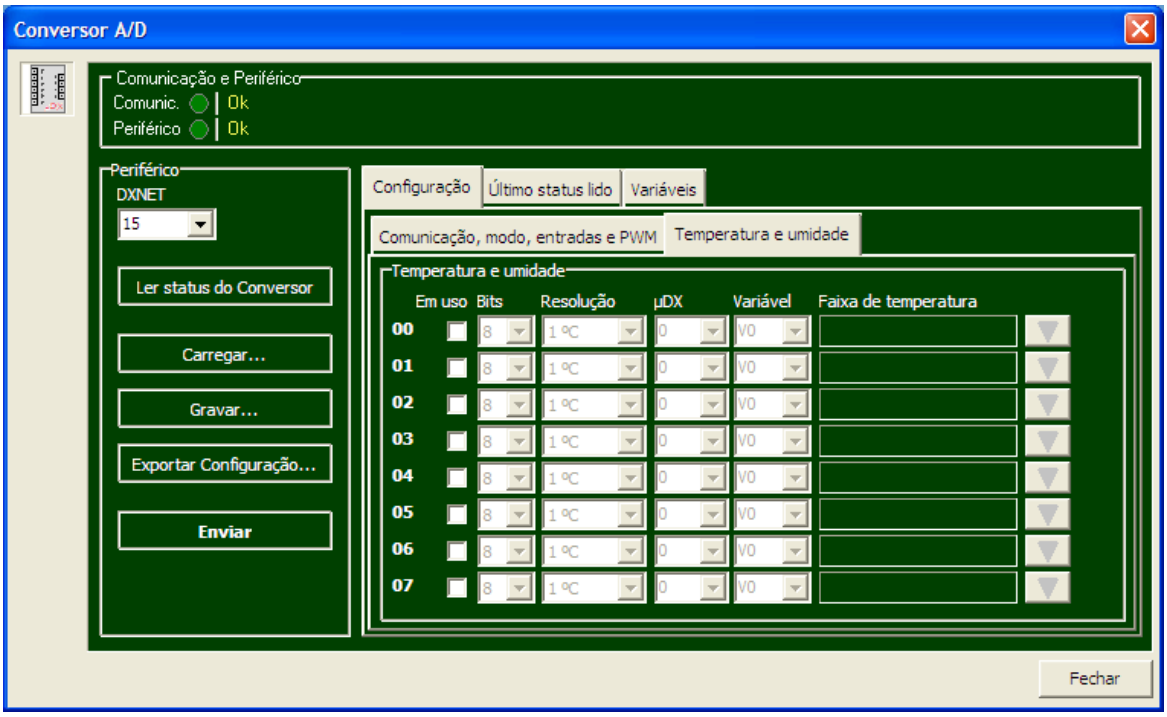
0 a 5 Vdc	→	resistência de entrada = 5 KΩ
0 a 5 Vdc	→	resistência de entrada ³ 1 MΩ (sem jumper)
0 a 10 Vdc	→	resistência de entrada = 10 KΩ
0 a 20 mA	→	resistência de entrada = 250 Ω

Além das 8 entradas, o conector de entradas analógicas do Conversor A/D possui uma ligação à fonte regulada de +5 Vdc e ao terra desta fonte. Isto permite conectar potenciômetros diretamente alimentados pelo conversor, por exemplo (neste caso, utiliza-se uma entrada programada para nível de entrada de 0 a 5 Vdc, sem nenhum jumper, para aumentar a resistência de entrada):



Note que a entrada utilizada com o potenciômetro não deve ter nenhum jumper instalado na placa impressa do conversor A/D. Assim, esta entrada irá ser de 0 a 5 Vdc com alta resistência de entrada, e a resistência do potenciômetro não irá influenciar a medida.

A aba seguinte é **Temperatura e umidade**, que permite configurar a leitura de sensores de temperatura e umidade conectados ao Conversor A/D:



Esta aba permite programar até 8 sensores (temperatura e umidade) a serem conectados ao Conversor A/D via rede I^2C. É possível conectar até 8 sensores de temperatura simultaneamente, ou 4 sensores de temperatura mais 4 sensores de umidade. Os sensores de temperatura podem ocupar os endereços 0 a 7 na rede I^2C, enquanto os sensores de umidade podem assumir apenas os endereços 4 a 7.

Sensores de Temperatura: a primeira opção em cada linha da tabela programa quais os sensores de temperatura ativos. A seguir, temos a especificação se os dados serão transmitidos em 8 ou 16 bits, qual a resolução, quais os endereços DXNET dos controladores µDX irão receber o valor de temperatura, a variável que irá receber o valor, e qual faixa de cada sensor de temperatura. Note que é possível conectar até 8 sensores de temperatura em um mesmo conversor A/D. Para isso a DEXTER disponibiliza placas de extensão (para fazer a derivação do cabo) e sensores de temperatura avulsos. O sensor de temperatura possui 3 “jumpers” (estrapes) que permitem programar seu endereço (de 0 a 7). Também é possível selecionar para que os

endereços de 4 a 7 recebam sensores de umidade, em vez dos sensores de temperatura. O sensor de umidade possui 2 “jumpers” (estrapes) que permitem programar seu endereço (de 4 a 7). Vamos examinar com mais cuidado as opções disponíveis nesta tabela:

Transmissão em 16 bits: Em 16 bits a variável selecionada do µDX100 irá receber o valor de temperatura inteiro (de °C em °C), e a variável subsequente o valor fracionário da temperatura. Neste exemplo, selecionamos a variável v0 do µDX endereço 5. Digamos que a temperatura do sensor 1 seja de 56,41 °C. Neste caso, a variável v0 do µDX 5 irá assumir valor 56, e a variável v1 irá assumir o valor fracionário (0,41). A representação é binária e funciona de forma similar aos números inteiros. Assim, o bit mais significativo de v1 terá valor 2-1 (0,5), enquanto o bit menos significativo de v1 terá valor 2-8 (0,00390625). Então, v1 irá assumir o valor:

0,41 x 2 = 0,82 → 0
0,82 x 2 = 1,64 → 1
0,64 x 2 = 1,28 → 1
0,28 x 2 = 0,56 → 0 ⇓
0,56 x 2 = 1,12 → 1
0,12 x 2 = 0,24 → 0
0,24 x 2 = 0,48 → 0
0,48 x 2 = 0.96 → 0

Logo, v1 irá assumir o valor 01101000 em binário, ou 104 em decimal. Note que 104 / 256 = 0,40625 » 0,41. Este método é similar a conversão de decimal para binário, no caso de números inteiros. Por exemplo, o valor de 56 de v0:

56 / 2 = 28 → 0
28 / 2 = 14 → 0
14 / 2 = 7 → 0
7 / 2 = 3,5 → 1 ⇓
3 / 2 = 1,5 → 1
1 / 2 = 0,5 → 1
0 / 2 = 0 → 0
0 / 2 = 0 → 0

Logo, v0 irá assumir o valor 00111000 em binário, ou 56 em decimal. Note que a dupla de variáveis que irá receber o valor de 16 bits da temperatura (no caso do exemplo, v0 e v1) tem os seguintes valores relativos, conforme seus bits:

Variável n :

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Variável n+1 :

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}

No caso de temperaturas negativas, é utilizada a notação em complemento de dois. Note que a máxima temperatura admissível pelo sensor é de 125°C. Como esta temperatura é menor que 128, significa que o bit mais significativo da variável (v0 no exemplo) será sempre 0 (zero) para temperaturas positivas e sempre 1 (um) para temperaturas negativas. Para obter a temperatura, no caso desta ser negativa (bit mais significativo ligado) basta inverter todos os bits e somar 1 ao resultado. Por exemplo, digamos que a temperatura detectada pelo sensor seja de -23,36 °C. Se esta temperatura fosse positiva, resultaria no valor 00010111 01011100 em binário (devido à limitação em 16 bits, o valor representado em binário é 23,359375°C em vez de 23,36°C). Para converter para negativo, basta inverter todos os bits e adicionar 1 ao resultado. Neste caso, ficaria:

Inversão:	11101000	10100011
Adição:	_____	+ 1
Valor negativo:	11101000	10100100

Logo, -23,36°C será representado em duas variáveis no µDX como 11101000 10100100. Note que o bit mais significativo está ligado. Para obter o valor de um número binário negativo

representado em complemento de dois basta seguir o caminho inverso. Inverta todos os bits e adicione 1 ao resultado para obter o valor positivo.

Por exemplo, para o valor obtido 11101000 10100100:

Inversão:	00010111 01011011
Adição:	_____ _____ + 1
Resultado:	00010111 01011100

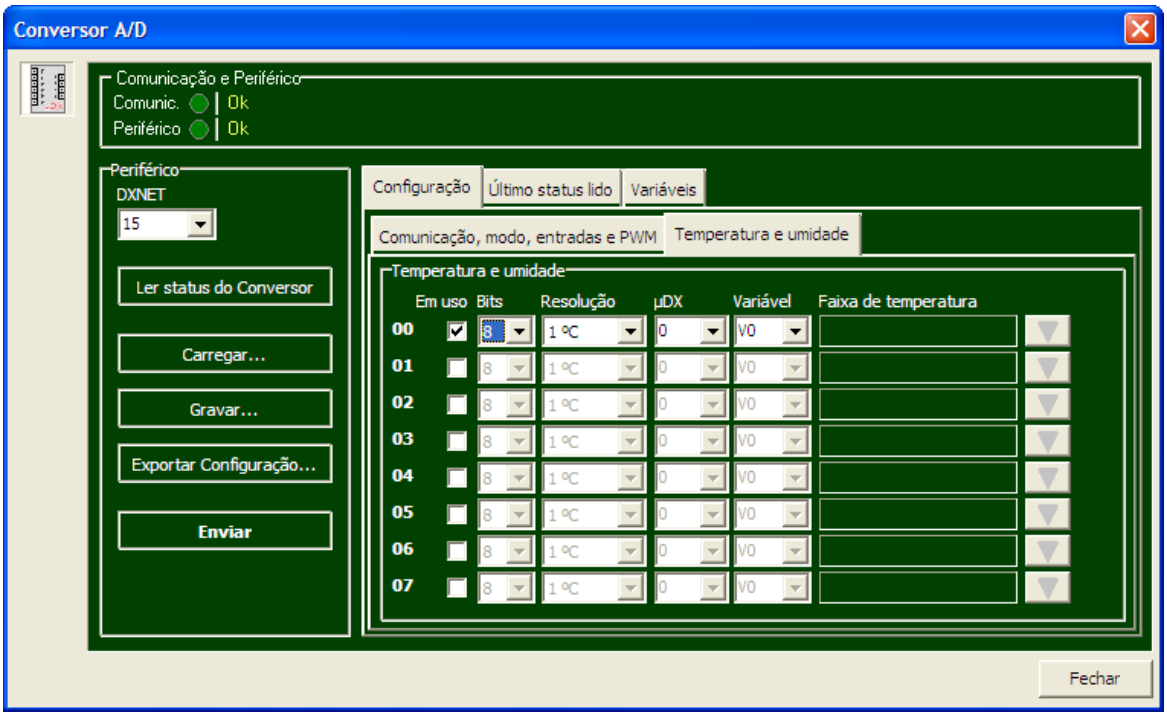
Todo este cálculo pode parecer complexo se o usuário não estiver familiarizado com álgebra booleana. Entretanto, nas aplicações usuais é utilizada a transmissão em 8 bits (explicada a seguir), que não utiliza representação de números negativos e necessita apenas de uma regra de três generalizada para obtenção da temperatura.

Transmissão em 8 bits: Já se selecionarmos transmissão em 8 bits, o programa PG irá requerer qual resolução e qual faixa a ser utilizada pelo sensor de temperatura. Estes dados serão utilizados pelo conversor A/D para compactar o valor de temperatura lido em 8 bits (tamanho de uma variável no µDX100). O sensor de temperatura permite ler temperaturas de -55°C a 125°C. Se formos ler temperatura de 1 em 1 °C, é possível abranger toda faixa do sensor em 8 bits. A temperatura de -55°C corresponderá ao valor 0 e a temperatura de 125°C ao valor 180 (125 + 55). No caso de transmissão em 8 bits não é utilizada representação de números negativos. A tabela na página seguinte indica a faixa de variação de temperatura passível de leitura em 8 bits para as várias resoluções disponíveis no PG.

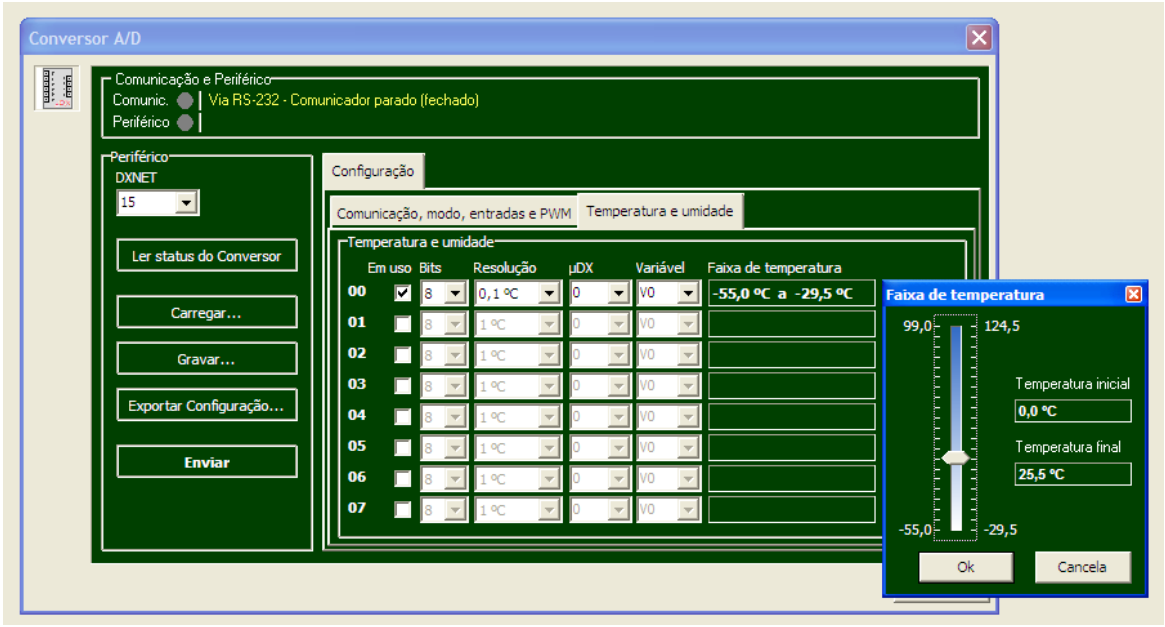
Resolução	**Faixa**
1 °C	180 °C
0,5 °C	127,5 °C
0,2 °C	51 °C
0,1 °C	25,5 °C
0,05 °C	12,75 °C
0,02 °C	5,1 °C

Ou seja, para uma resolução de 1 °C pode-se usar a faixa completa de medição do sensor (de -55°C a 125°C ® 180°C). Já se a resolução especificada for de 0,5 °C, a faixa é de 127,5°C (pode-se ler de -55°C a 72,5°C; ou de -2,5°C a 125°C, por exemplo). Se a resolução for maior, como 0,1 °C, a faixa diminui proporcionalmente (com resolução de 0,1 °C a faixa é de apenas 25,5°C - já que em 8 bits pode-se representar um valor máximo de 255). Para uma resolução de 0,1 °C, pode-se especificar uma faixa de medição de 15°C a 40,5°C, por exemplo. Neste caso, o valor 0 (zero) na variável do µDX indicará temperatura de 15°C e o valor 255 indicará temperatura de 40,5°C.

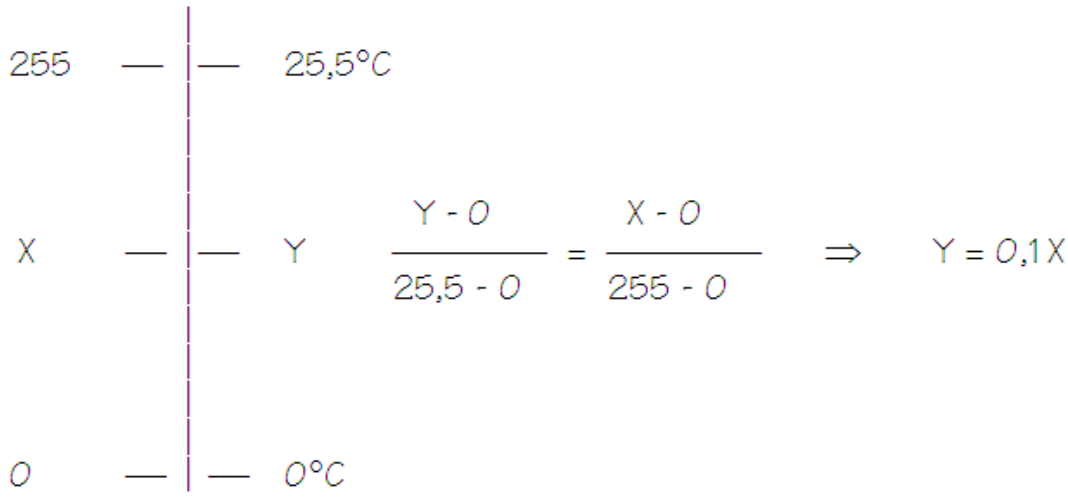
Vamos prosseguir com um exemplo. Selecionando o modo de 8 bits surge a especificação de resolução do sensor:



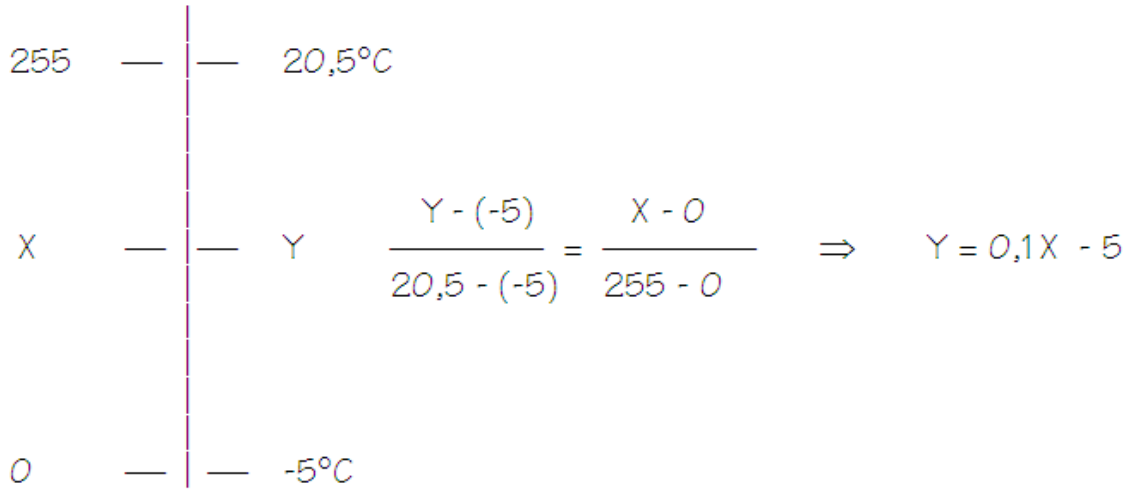
Neste exemplo irei utilizar resolução de 0,1 °C. Uma vez especificada a resolução o próximo dado requerido é a faixa do sensor de temperatura (ou seja, entre que valores de temperatura o sensor irá atuar). Note que o programa PG já calcula a largura da faixa conforme a resolução especificada pelo usuário. Assim, basta selecionar a temperatura mínima a ser lida (a temperatura máxima é calculada pelo PG). No exemplo selecionei 0 °C como temperatura mínima (resultando uma faixa de 0 °C a 25,5 °C).



Como em 8 bits não é usada a representação de números negativos, a temperatura mínima da faixa selecionada sempre corresponde ao valor 0 da variável do µDX100, enquanto a temperatura máxima da faixa selecionada corresponde ao valor 255 da variável do µDX100. No exemplo, foi usada resolução de 0,1 °C e faixa de 0 °C a 25,5 °C. Neste caso:



Portanto, para obter a temperatura basta ler o valor da variável do µDX100 e dividir por 10. Se a faixa escolhida fosse de -5°C a 20,5°C, para a mesma resolução de 0,1°C, teríamos:



Ou seja, para obter a temperatura basta dividir o valor lido na variável (X) por 10 e subtrair 5. Se for lido valor 78, por exemplo, corresponderá a uma temperatura de 0,1 x 78 - 5 = 2,8 °C. Com isso pode-se deduzir uma fórmula geral:

Temperatura = Resolução x Variável + Temp.Inicial

Ou seja, para calcular a temperatura multiplica-se a resolução pelo valor da variável do µDX100 e soma-se a temperatura inicial da faixa programada para o sensor.

Sensores de Umidade: No caso dos endereços de 4 a 7, além das resoluções de 1°C a 0,02°C para os sensores de temperatura, surge uma opção adicional com a seguinte inscrição: 0,5% UR. Ao selecionar esta opção programa-se este endereço para leitura de sensor de umidade, com resolução de 0,5 % de umidade relativa (UR). Note que os sensores de umidade vêm de fábrica programados para o endereço 4, e os sensores de temperatura para o endereço 0.

No caso do sensor de umidade, a resolução é fixa em 0,5% UR. Assim, para obter-se a umidade relativa do ambiente basta dividir por 2 o valor da variável que recebe o valor lido pelo sensor. Para 0% de UR esta variável irá assumir valor 0, enquanto que para 100% de UR a variável assumirá valor 200. Para uma umidade relativa de 56,5 % UR, a variável será 113.

Tanto o sensor de temperatura (se programado para leitura em 8 bits) quanto o sensor de umidade retornam valor 0 na respectiva variável, caso sejam desconectados do Conversor A/D. Isso facilita a detecção de problemas na instalação, como rompimento da cablagem dos sensores.

Envia senha de comunicação

Caso o programa PG esteja utilizando comunicação serial RS232 e o Modem para µDX100 esteja programado para aguardar senha pela serial é preciso enviá-la para iniciar a comunicação de dados. Esta senha é especificada na Configuração do Comunicador (menu pop-down **Comunicação**), aba **Modem**.

Executar Programa

Ao transmitir o programa aplicativo para o controlador µDX100 este assume o estado parado, por segurança. Para executar o programa é necessário enviar este comando.

Parar Execução do Programa

Esta opção permite parar o programa aplicativo do controlador µDX100.

Solicitar Status

Este comando solicita envio do status (várias informações, como versão de firmware, ciclo de execução do programa, endereço DXNET, etc.) do controlador µDX100. Esta informação é necessária para iniciar monitoramento do CLP.

Reset (soft)

Efetua um reset no µDX100. O programa aplicativo continua no estado que estava anteriormente ao reset (parado ou executando), o relógio de tempo real é preservado, mas as demais variáveis são reinicializadas para valor 255.

Limpa todos os nodos forçados

O forçamento de nodos em estado ligado via programa PG atua em **nodos_DX**. Estes nodos só são acessíveis por rede DXNET e, portanto, só são atuados pelo PG ou por outro controlador µDX100 na rede DXNET (via blocos DXNET no programa aplicativo). Os **nodos_DX** e os **nodos_OUT** (gerados pelo programa aplicativo em execução no µDX100) são combinados via operação OR (OU) para gerar o estado final dos nodos. Assim, se o nodo já está forçado em estado ligado pelo programa aplicativo não é possível desligá-lo via PG, pois o resultado de qualquer operação OR com um dos operandos em 1 é 1. Já se o programa aplicativo não está acionando determinado nodo é possível acioná-lo via PG. Entretanto, uma vez acionado via PG este nodo ficará em 1 até que o PG (ou outro µDX100 na rede DXNET) mande desligá-lo. Então, esta opção do menu **µDX** permite desligar todos os nodos que foram forçados via programa PG, fazendo com que os mesmos possam ser controlados pelo programa aplicativo no µDX100.

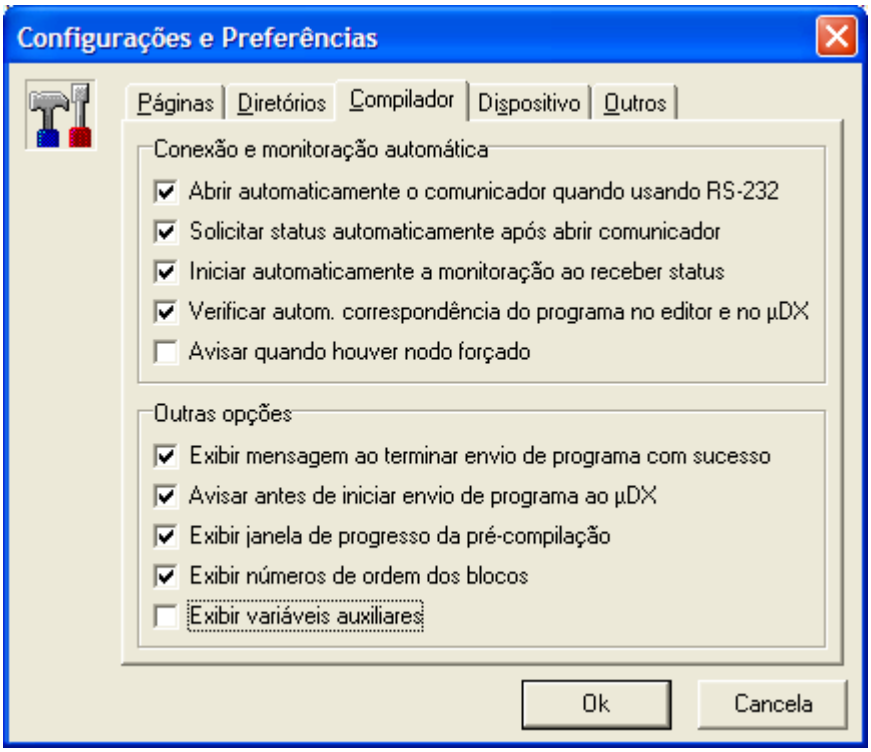
Iniciar Monitoração

Inicia a monitoração de status, nodos e variáveis do µDX100. Note que o dado **Refresh**, existente na tela do Compilador PG (aba Status+CPU+Programa+Cfg.Hardware), indica o tempo total para uma leitura completa de todos os dados. Evidentemente, quanto maior o número de variáveis e nodos a serem monitorados mais lenta fica a monitoração. Também influencia este tempo o meio de comunicação empregado (porta serial, endereço TCP/IP, ou ainda Modem). Caso exista programa aplicativo carregado no Compilador PG (sufixo .u1p), e este seja idêntico ao carregado no µDX100 sob monitoração (para isso é preciso pressionar a opção **Verifica programa no µDX**), todas as variáveis e nodos empregados no programa aplicativo são monitorados.

Para iniciar monitoração é preciso enviar senha (caso o modem o exija para comunicação serial) e solicitar status. Caso em **Configurações** → **Configurações e Preferências** → **Compilador** estejam marcados os itens:

[x] Abrir automaticamente o comunicador quando usando RS-232
[x] Solicitar status automaticamente após abrir comunicador.
[x] Iniciar automaticamente a monitoração ao receber status.
[x] Verificar autom. correspondência do programa no editor e no µDX.

todo o procedimento é automático (esta é a condição default ao instalar o PG).

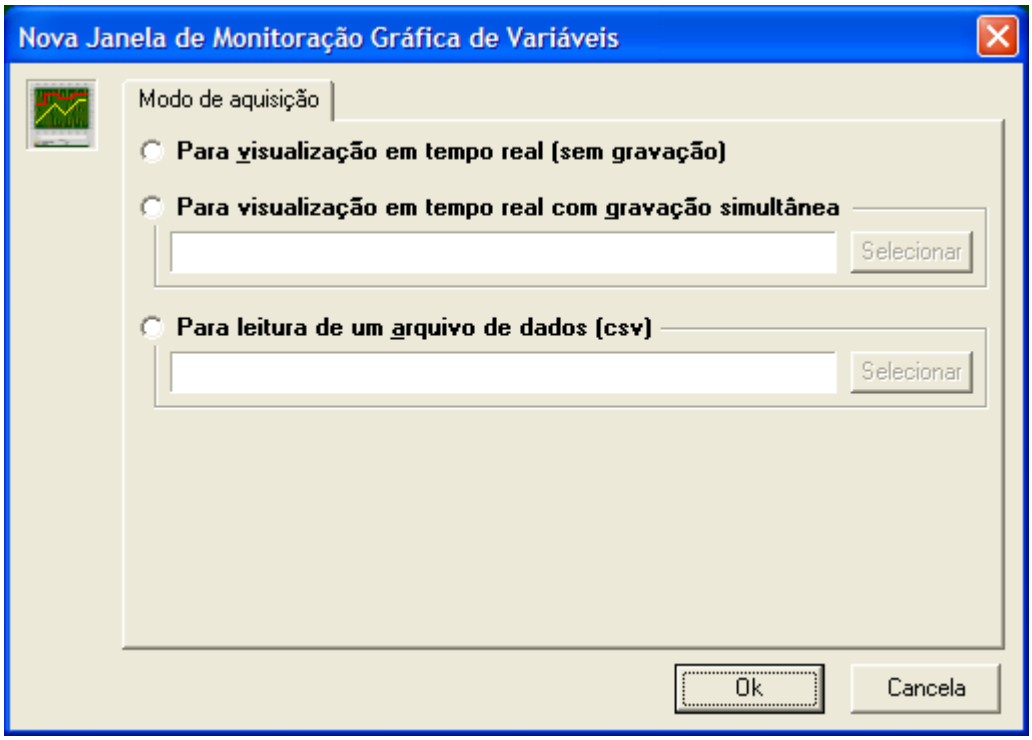


Parar Monitoração

Interrompe a monitoração. Isso pode ser importante para aliviar o processamento do computador, caso se queira usá-lo para outros fins.

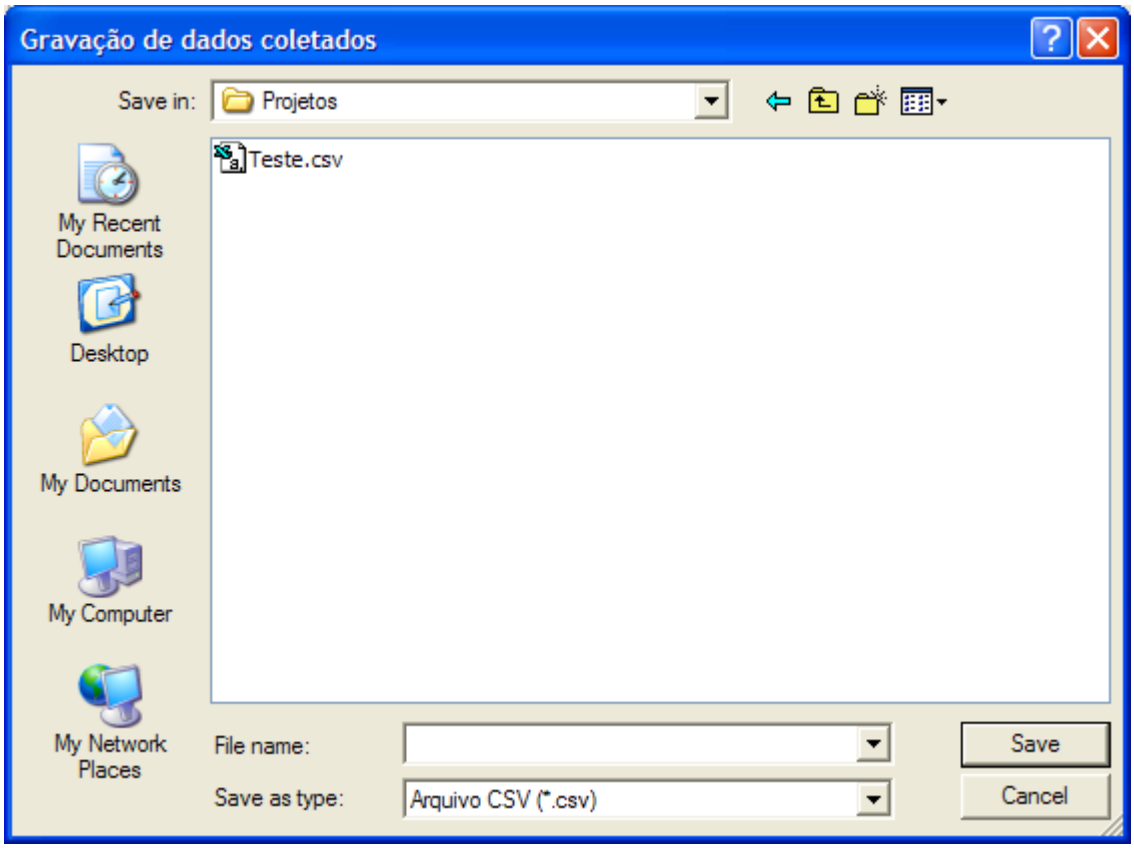
Nova Janela de Monitoração...

Esta opção cria uma nova janela de monitoração gráfica. Com isso, é possível monitorar graficamente variáveis do Controlador µDX100, e salvar os valores lidos em arquivos compatíveis com planilhas eletrônicas como Excel. Inicialmente é criada uma janela com as seguintes possibilidades:

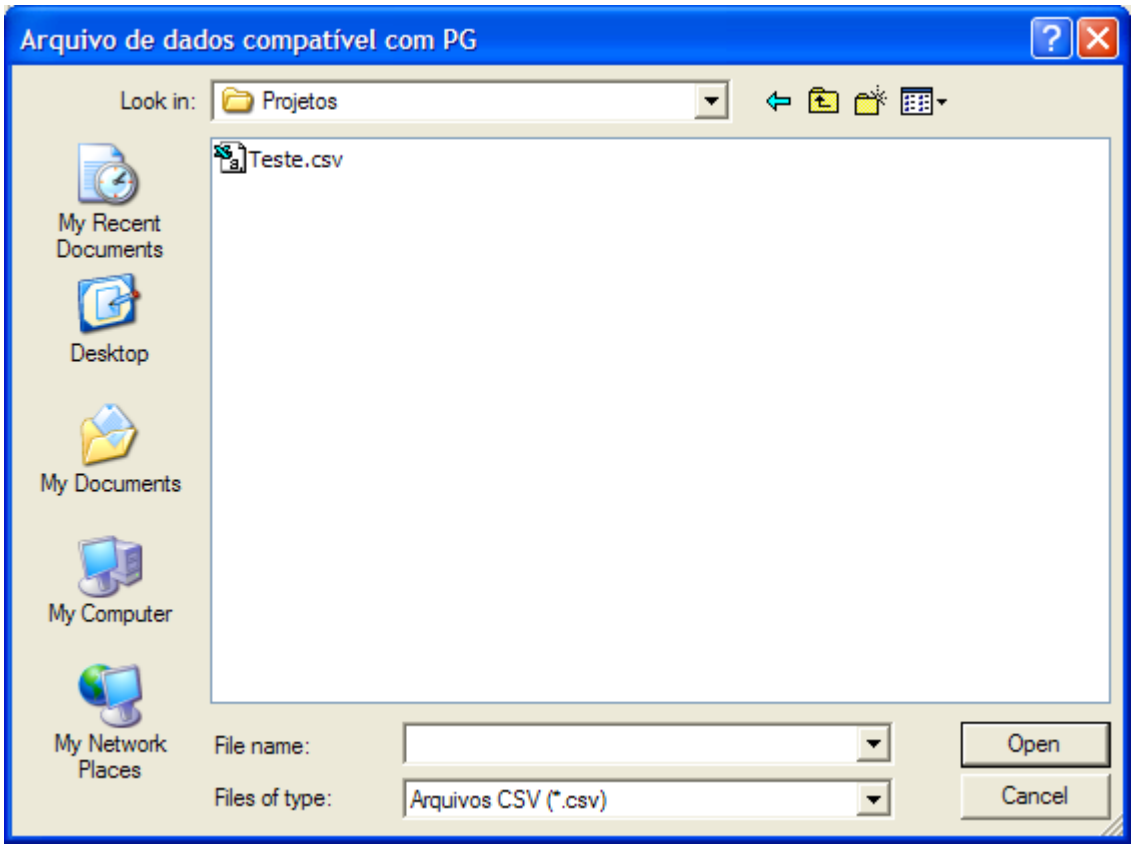


O **Modo de Aquisição** pode ser de três tipos:

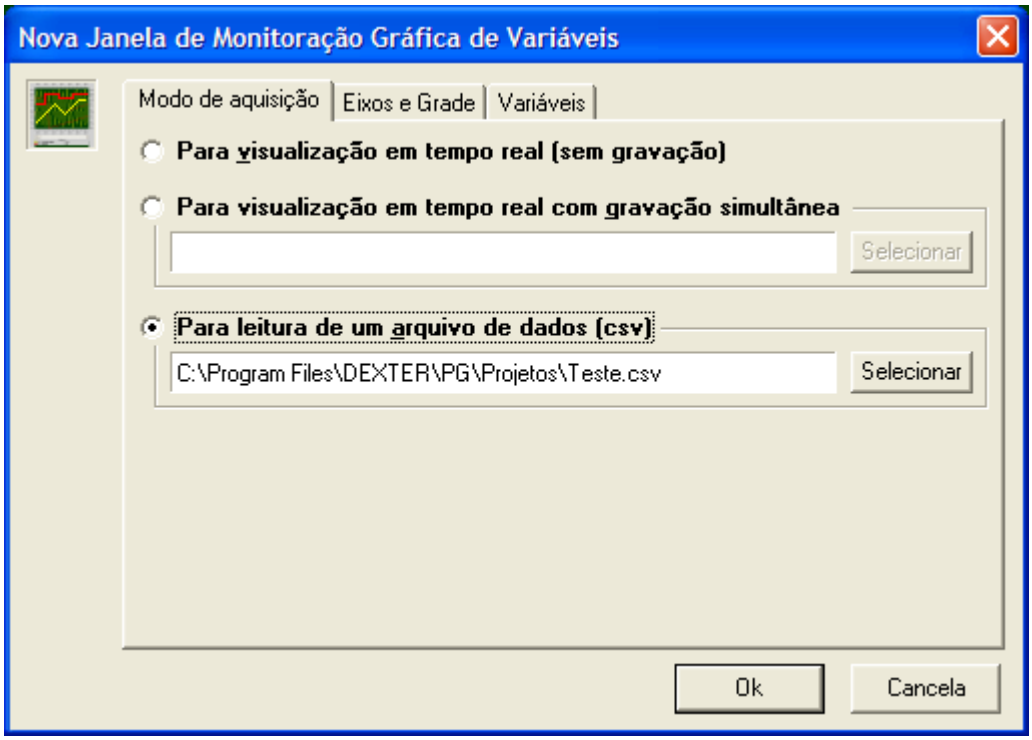
- **Para visualização em tempo real (sem gravação):** neste caso os dados irão aparecer na tela em tempo real, mas não serão gravados. Evidentemente, neste caso está descartado poder navegar para áreas anteriores do gráfico (ao completar a tela disponível todo o gráfico passa a rodar horizontalmente, descartando os dados mais antigos).
- **Para visualização em tempo real com gravação simultânea:** idêntico ao caso anterior, mas os dados lidos são também gravados em arquivo de tipo CSV, passível de ser lido posteriormente tanto no próprio PG como em uma planilha eletrônica como Excel, por exemplo. Ao selecionar esta opção surge uma janela para seleção do nome do arquivo a ser gerado com os dados:



- **Para leitura de um arquivo de dados (CSV):** permite ler um arquivo do tipo CSV gravado anteriormente via opção anterior gravado com dados do µDX100. Neste caso surge uma janela para seleção do arquivo a ser lido:

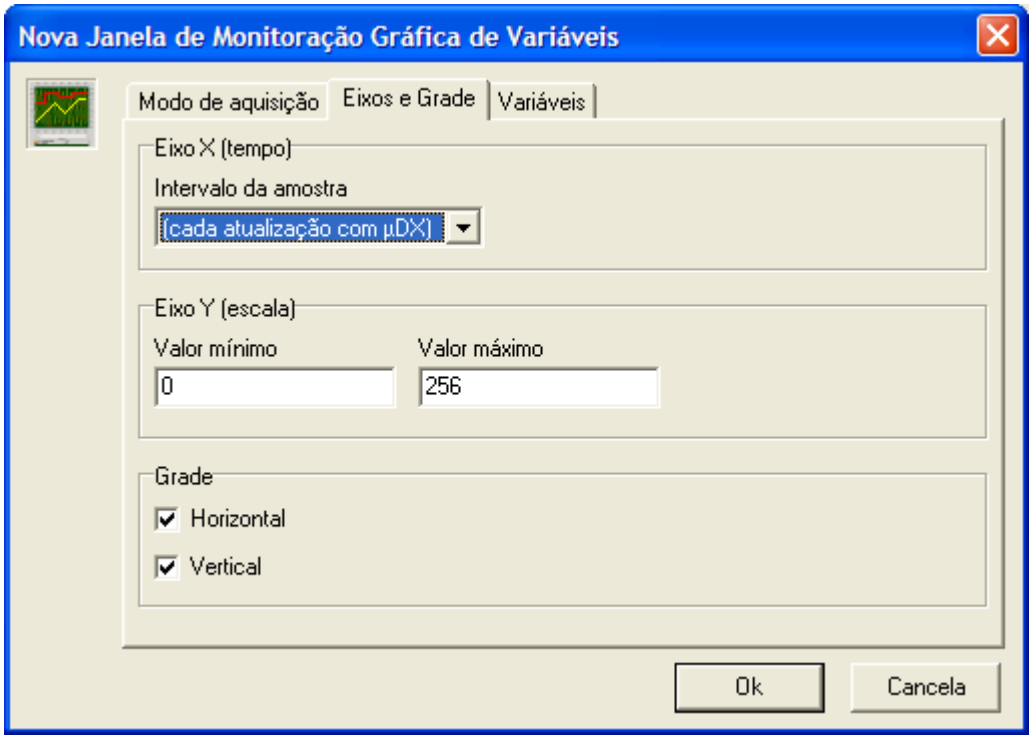


Ao selecionar uma das opções acima duas outras abas são ativadas na janela **Nova janela de Monitoração Gráfica de Variáveis**. São elas **Eixos e Grade** e **Variáveis**.

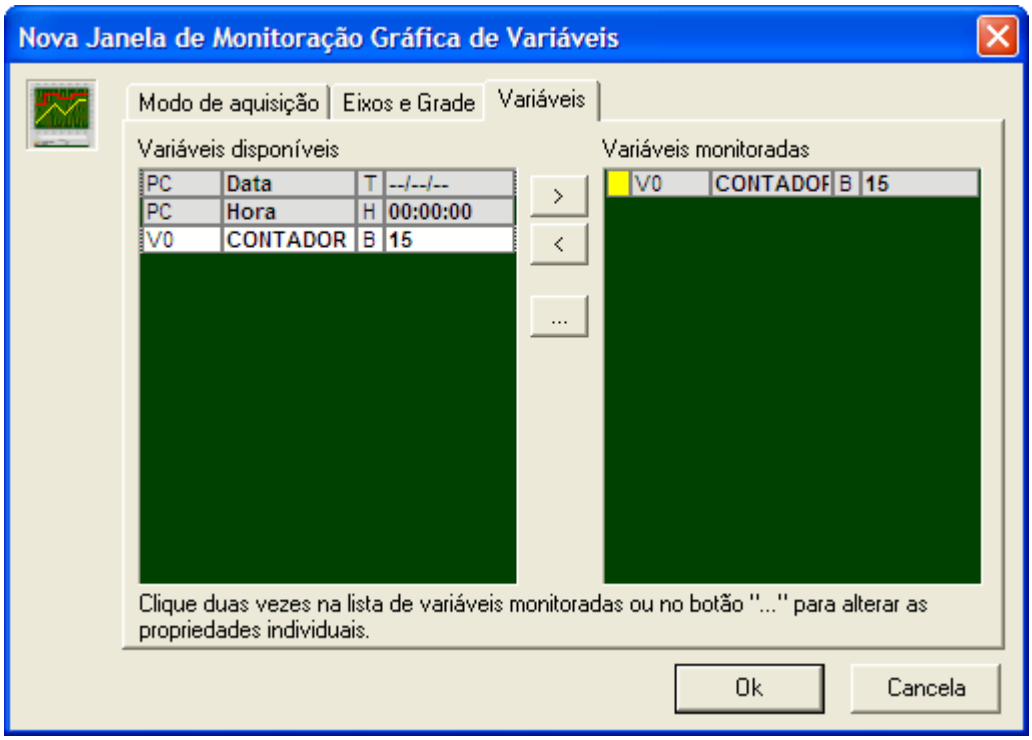


Na aba **Eixos e Grade** é possível selecionar o valor mínimo e máximo do eixo Y do gráfico (o

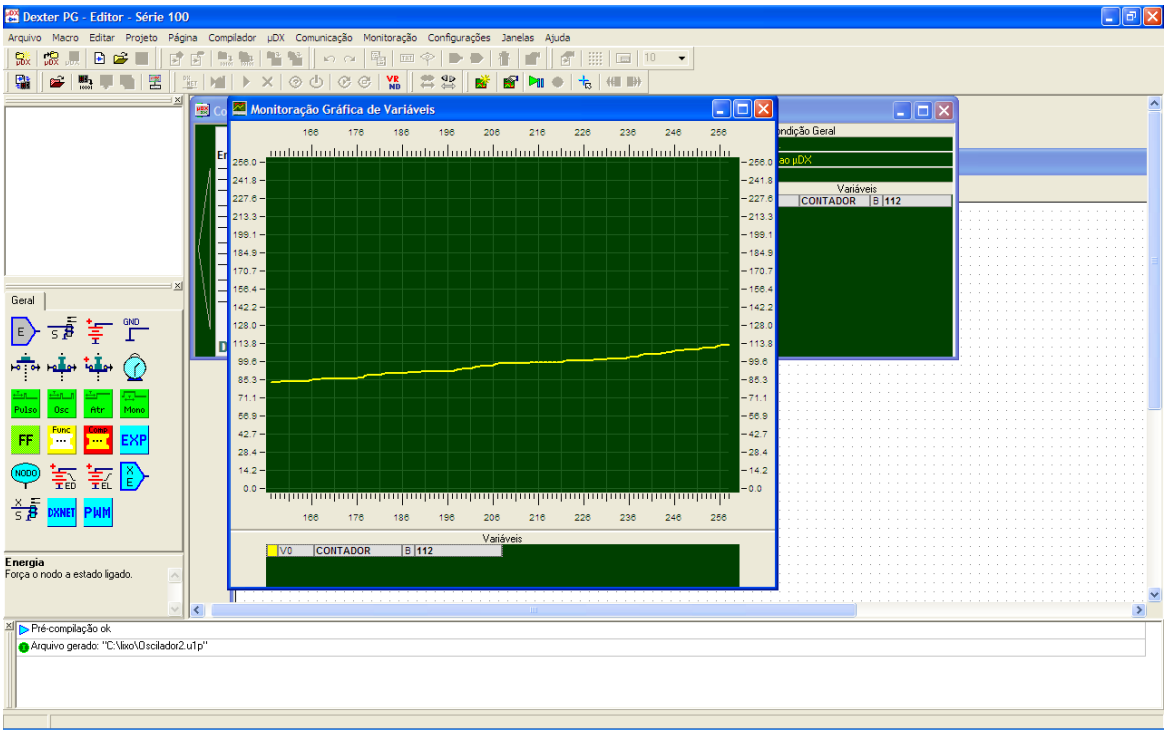
valor default é 0 e 256), o intervalo de amostra do eixo X (o default é a cada atualização com µDX, ou seja, a velocidade máxima de monitoramento do PG). Por fim, também pode-se inibir a grade horizontal ou vertical do gráfico.



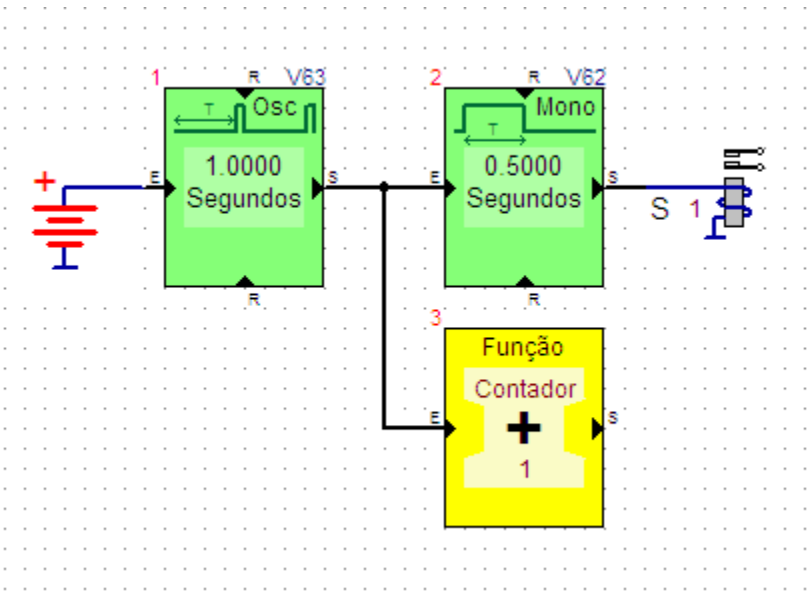
Na aba **Variáveis** seleciona-se quais variáveis serão plotadas no gráfico. Para que apareçam variáveis do µDX nas possibilidades de seleção desta janela é preciso que o Compilador PG esteja com a monitoração ativada (se não somente data e hora do microcomputador PC estarão disponíveis). Para selecionar as variáveis a serem plotadas basta clicar duas vezes sobre as mesmas, ou selecioná-las e clicar no botão  . O botão  permite eliminar variáveis da plotagem. O botão  possibilita editar o nome da variável e sua cor.



Caso o programa aplicativo existente no µDX100 coincida com o programa aplicativo carregado no Compilador PG (verificável via opção **Verifica programa no µDX**) estarão disponíveis para plotagem todas as variáveis usadas no programa do CLP. Ao clicar em Ok é gerada a janela de monitoração gráfica. A janela de monitoração gráfica a seguir foi gerada com captura da variável CONTADOR, que é incrementada a cada segundo:



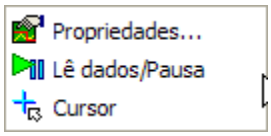
O programa aplicativo que gerou esta variável é:



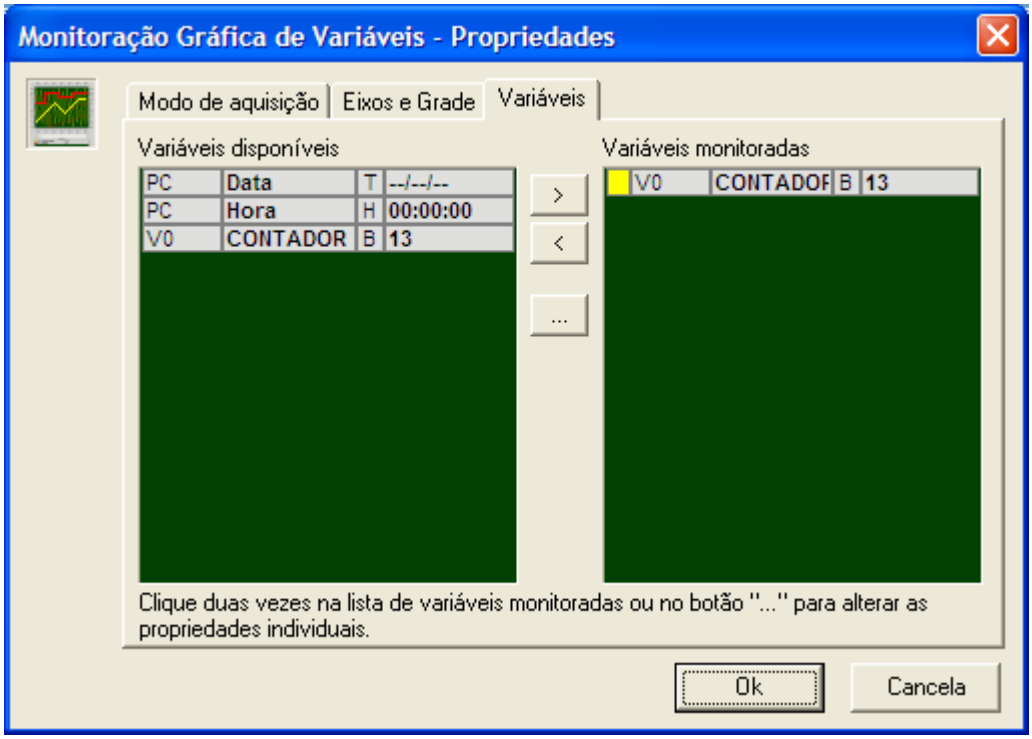
Exemplo de Programa Aplicativo: **Oscilador2.d1g** (fonte); **Oscilador2.u1p** (compilado).

Monitoração Gráfica

Permite selecionar diversas funções para a monitoração gráfica de variáveis. São elas:

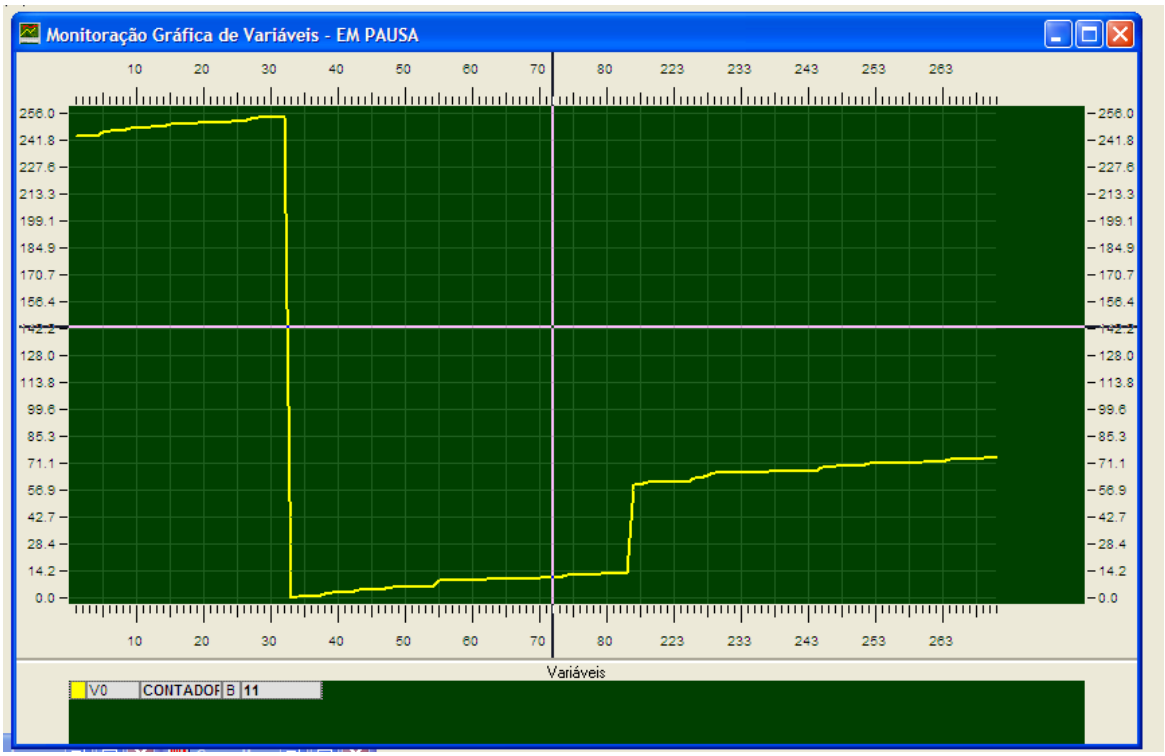


Propriedades: Acessa os parâmetros do gráfico, como escala X e Y, variáveis monitoradas, etc., permitindo modificá-los.



Lê dados/Pausa: interrompe ou reinicia a leitura de dados do gráfico.

Cursor: cria um cursor sobre o gráfico, permitindo a leitura dos valores das variáveis em qualquer ponto do mesmo.

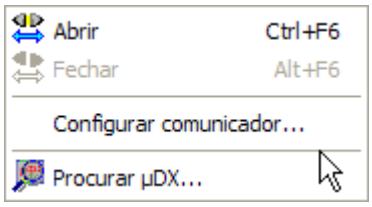


Acertar Relógio...

Permite acertar o relógio de tempo real do controlador µDX100. É possível atribuir a data e hora existentes no relógio interno do computador ligado ao µDX100 ao relógio de tempo real desse. Note que o µDX100 permite apenas a inclusão de dia da semana na data. Também é possível especificar a hora manualmente no relógio interno do µDX100. Esta opção pode ser útil para teste de programas com comportamento dependente do relógio de tempo real do controlador.

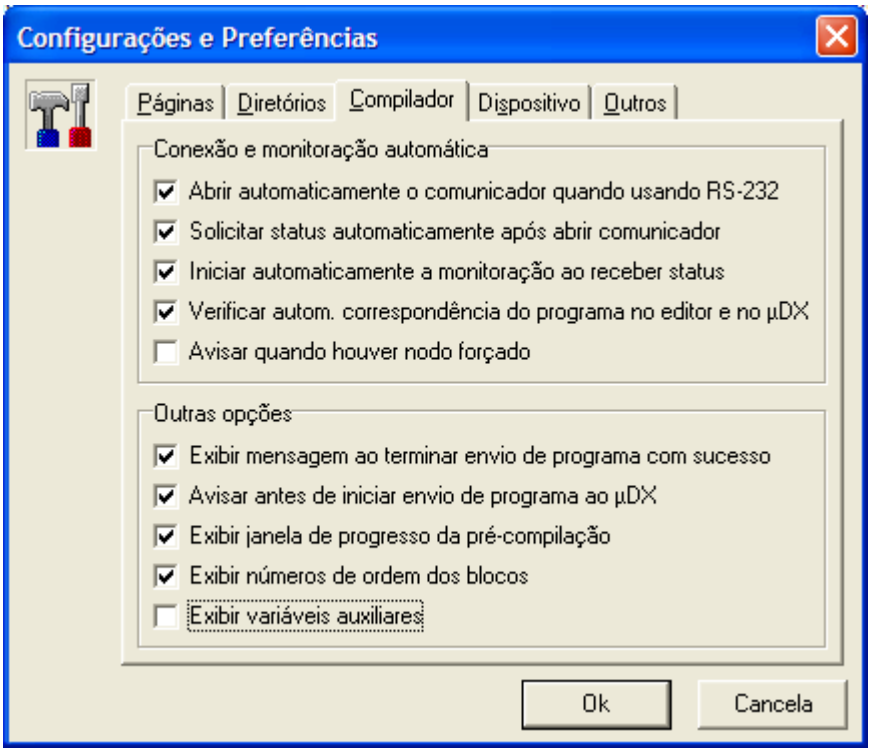
Menu Comunicação

O menu Comunicação concentra as opções de comunicação com o controlador µDX100:



Abrir (Ctrl+F6)

Abre o comunicador. Com isso, o Compilador PG aloca o recurso para si (porta serial no caso de comunicação serial; endereço TCP/IP no caso de comunicação via rede Ethernet; ou modem no caso de comunicação via linha telefônica). Caso o comunicador esteja utilizando comunicação serial e as opções automáticas estejam habilitadas no menu **Configurações → Configurações e Preferências → Compilador** é solicitado o status do CLP e é iniciada a monitoração.



Fechar (Alt+F6)

Fecha o comunicador, liberando o recurso para que outros programas possam utilizá-lo (porta serial no caso de comunicação serial; endereço TCP/IP no caso de comunicação via rede Ethernet; ou modem no caso de comunicação via linha telefônica).

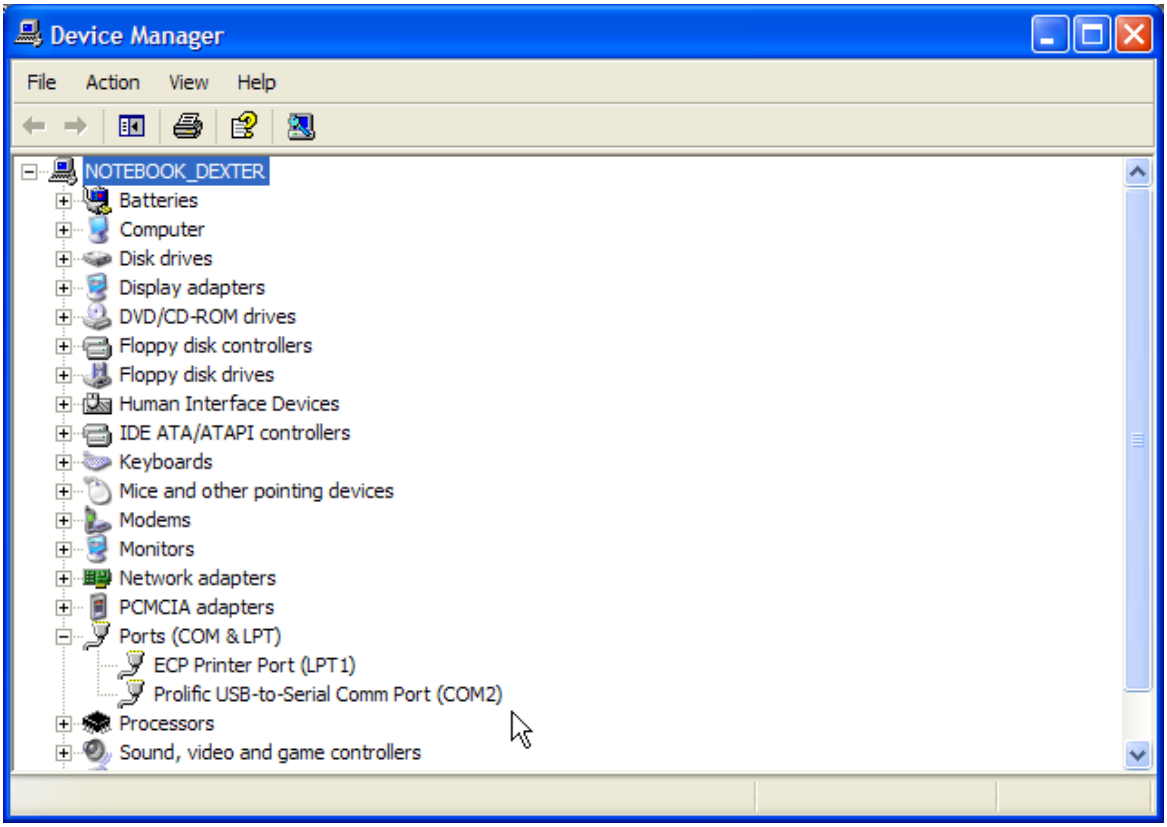
Configurar Comunicador...

A **Configuração do Comunicador** possui cinco abas. A primeira, **Comunicador**, permite escolher qual a forma de comunicação com o controlador µDX100. As opções são comunicação serial RS-232, comunicação via rede Ethernet, ou ainda via Modem. Nesta aba também existe um campo para determinar o timeout mínimo de comunicação (o default é 500ms). Este timeout é usado caso os timeouts programados para cada tipo de comunicação sejam menores, ou seja, determina o menor tempo de espera de respostas à perguntas feitas pelo PG. Por fim, é possível especificar qual protocolo será usado pelo programa PG (DXNET ou DXNET+). Evidentemente ele deve coincidir com o protocolo escolhido no Modem para µDX100 ligado ao PG.

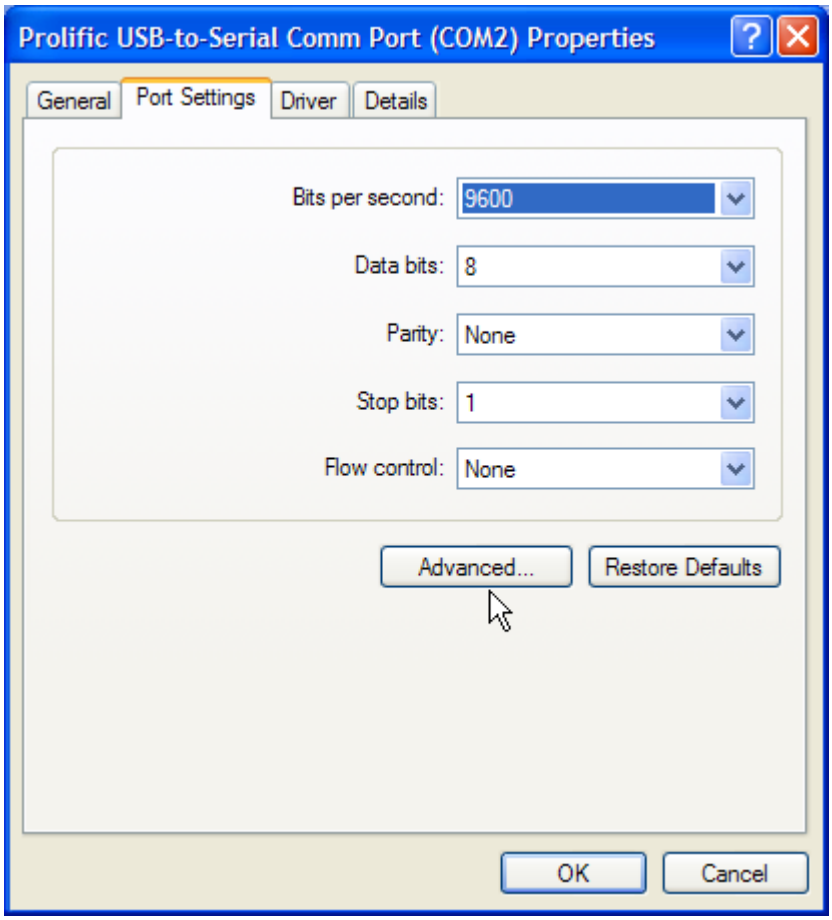
A aba **RS-232** configura a comunicação serial com o µDX100. O configuração default do controlador é 9600 bps, sem paridade, 8 bits de dados, 2 stop bits. Existe um campo para timeout da comunicação RS-232. Este timeout será usado para este tipo de comunicação caso ele seja superior ao timeout mínimo geral (programado na aba **Comunicador**). Note que em 9600 bps o campo de stop bits é desabilitado, uma vez que nesta velocidade o Modem para µDX100 necessita de dois stop bits.

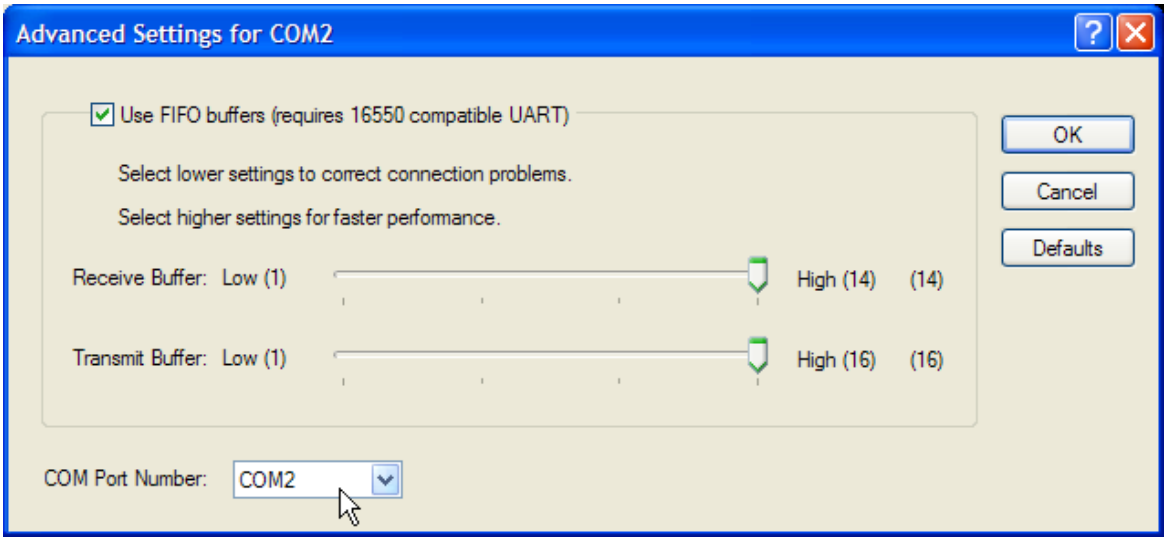
Atenção: caso seu computador não possua porta serial é possível utilizar um cabo adaptador USB↔RS-232. A Dexter comercializa este tipo de cabo, devidamente testado com o µDX100. Solicite um orçamento. De forma geral, todos os cabos adaptadores USB↔RS-232 do mercado devem funcionar sem problemas com o controlador µDX100.

Nos casos de inexistência de porta serial RS232C pode-se usar um cabo USB↔RS-232. Normalmente acompanha o cabo USB↔RS-232 um CD com o driver para Windows. Ele deve ser instalado para o correto funcionamento do dispositivo (com alguns cabos isso é desnecessário, e o Windows reconhece o cabo automaticamente). Para saber qual a porta de comunicação que foi gerada ao conectar o cabo adaptador USB↔RS-232 vá em **Painel de Controle** → **Sistema** → **Hardware** → **Gerenciador de Dispositivos** → **Portas (COM & LPT)** e verifique qual a COM gerada:



Caso se queira modificar o valor da COM clique duas vezes sobre o driver para porta serial no Gerenciador de Dispositivos (Device Manager) para abrir as propriedades da porta serial e selecione a opção **Avançado** na aba **Configurações de Porta**:





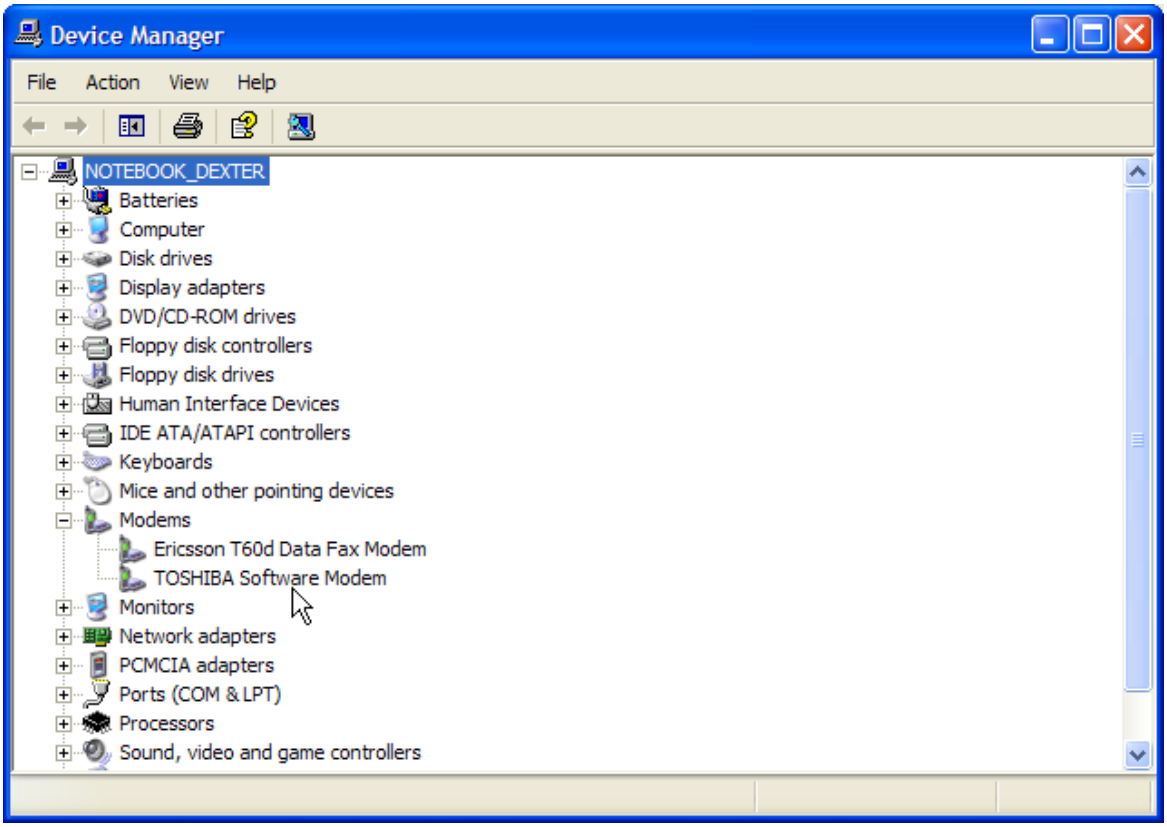
A aba **Modem** especifica dados necessários para estabelecer comunicação serial (ou via rede telefônica) com Modem para µDX100. Como o controlador não possui porta serial RS232C este equipamento deve ser intercalado entre a rede DXNET do controlador e a porta serial do computador. Temos o endereço do Modem para µDX100 na rede DXNET, a senha (que só é importante se o Modem estiver programado para exigir senha em comunicação serial) e o código alfanumérico (só é importante em conexões via rede telefônica discada).

A aba **TCP/IP** permite selecionar o endereço do controlador µDX100 na rede Ethernet, o número da porta utilizado, e a velocidade de comunicação serial usada entre o CLP e o adaptador Ethernet↔RS-232. Note que é preciso intercalar entre o Modem para µDX100 e a rede Ethernet um adaptador Ethernet↔RS-232 (dispositivo facilmente obtenível no mercado. A Dexter pode fornecer tal equipamento se necessário, ou indicar fornecedores).

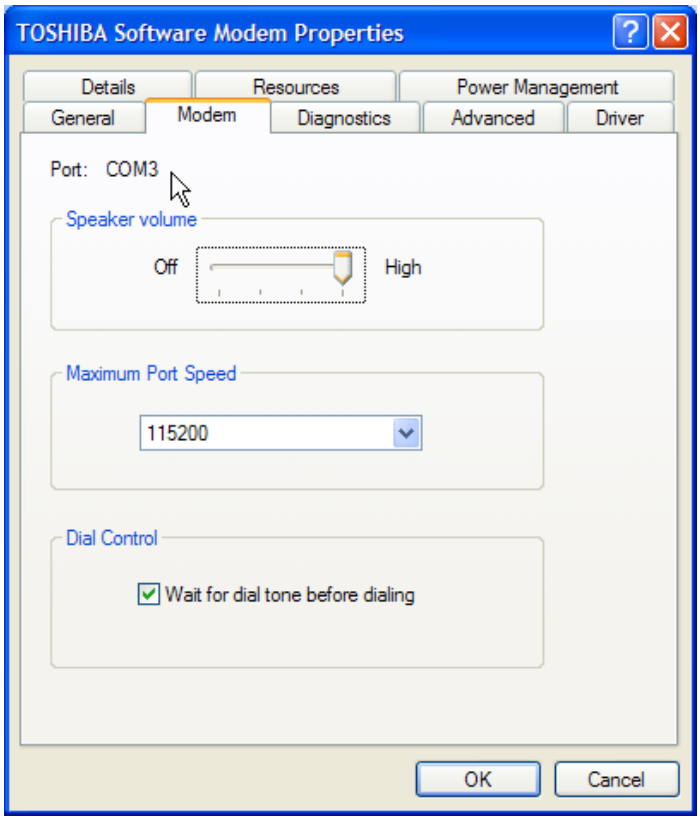
Por fim, a aba **Modem telefone** configura a comunicação via rede telefônica fixa. Note que a comunicação é direcionada para uma porta serial, onde deve existir um modem (interno ou externo ao computador). Evidentemente, do outro lado da comunicação telefônica é mandatório existir um modem conectado via rede DXNET ao µDX100.

Temos os campos para seleção da porta de comunicação serial ligada ao modem, velocidade de comunicação (em bits por segundo), paridade e número de stop bits. Também é possível programar um timeout para este tipo de comunicação (caso este dado seja menor que o timeout mínimo geral programado na aba **Comunicador** prevalecerá o timeout mínimo geral). O baud rate aceito pelo Modem para µDX100 em comunicação via rede telefônica é apenas o de 300 bps (Bell 103).

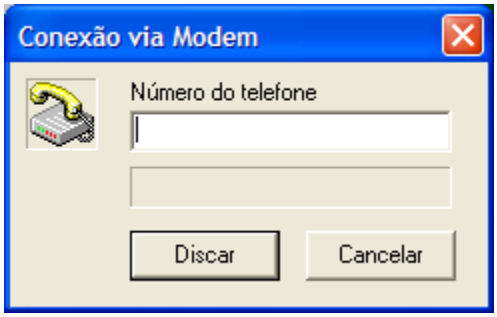
Existe um campo para programar string de inicialização do modem, cujo valor inicial é ATE0V1. Para descobrir qual a porta serial onde está o modem do computador usa-se, novamente, a janela **Gerenciador de Dispositivos** no sistema operacional Windows, selecionando o item **Modems**:



Clique duas vezes sobre o driver para modem de forma a abrir as propriedades do mesmo, e vá na aba **Modem** para obter a porta COM ocupa pelo modem do computador:

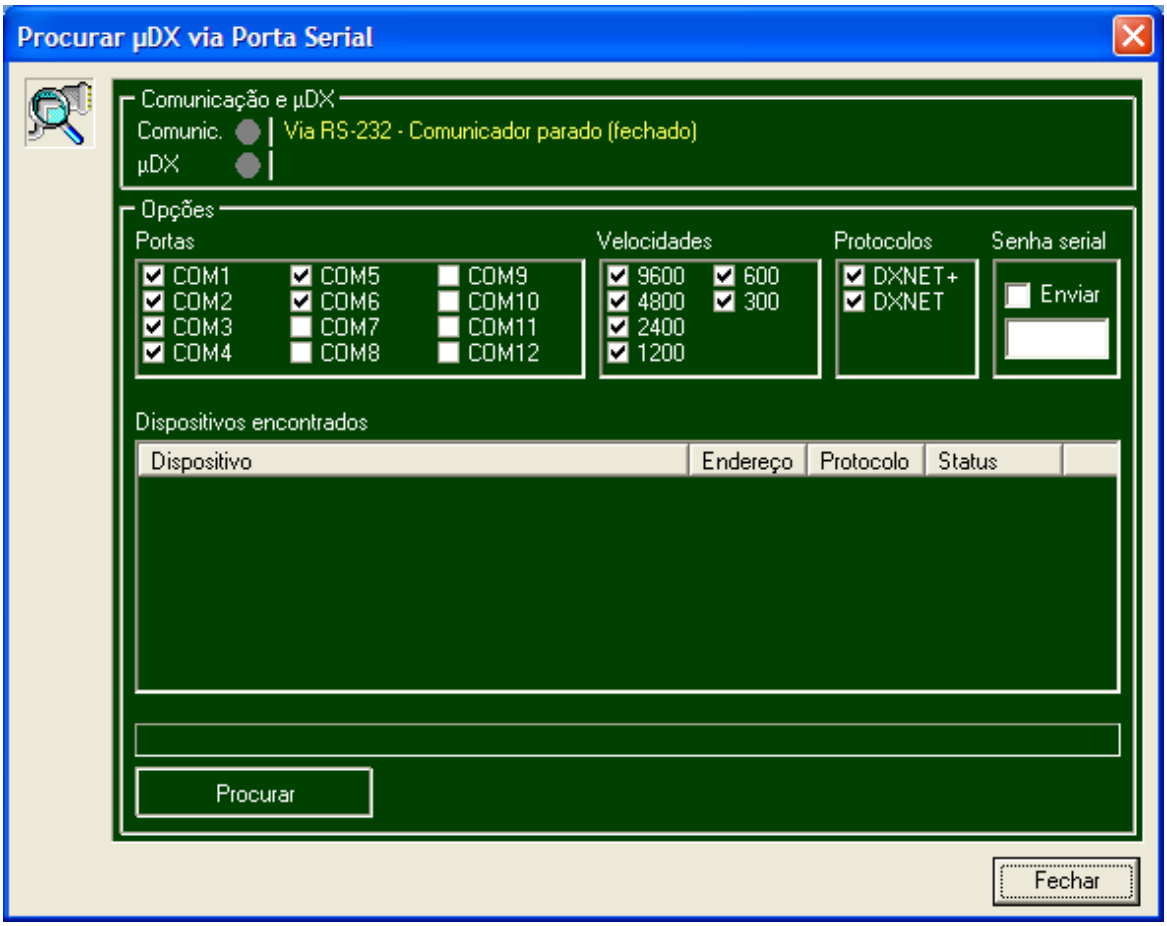


Caso esteja selecionada a opção Modem como meio de comunicação, ao abrir Comunicador do PG surgirá a tela seguinte, de forma a ser informado o número de telefone para discagem:



Procurar µDX...

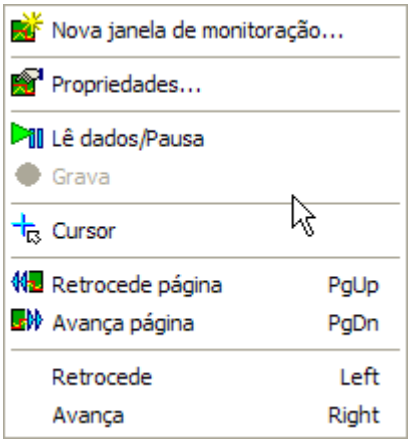
Devido a grande quantidade de opções de velocidade de comunicação, e as diversas portas seriais que podem estar disponíveis no computador, pode ser bastante trabalhoso descobrir manualmente a configuração do Modem para µDX100 para comunicação RS-232. Neste caso esta ferramenta é bastante útil. Ela tenta estabelecer comunicação serial com o CLP em todas as portas COM selecionadas, e em todos os baud rates e configurações escolhidos:



Clique em Procurar para iniciar a varredura em todas as portas seriais (COM) selecionadas.

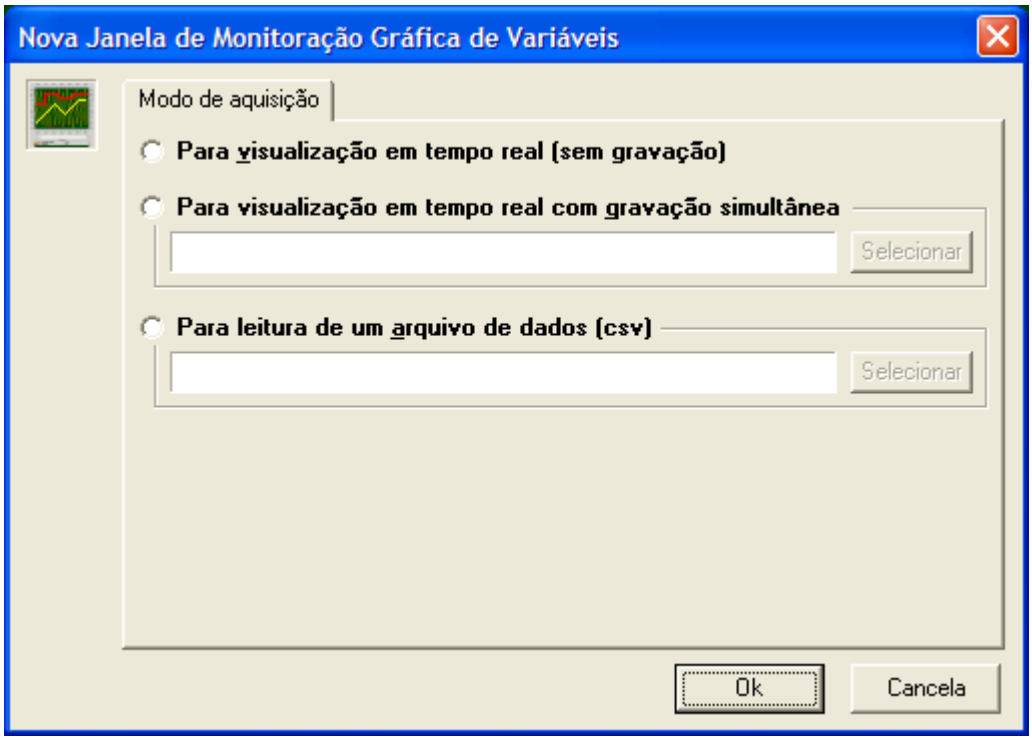
Menu Monitoração

Este menu controla as opções para monitoração gráfica de variáveis no Compilador PG. Com isso, é possível capturar valores de variáveis e plotá-las em um gráfico, facilitando a depuração de algoritmos de controle analógicos e leitura de sensores, por exemplo. É possível abrir diversas janelas de monitoração simultaneamente no PG.



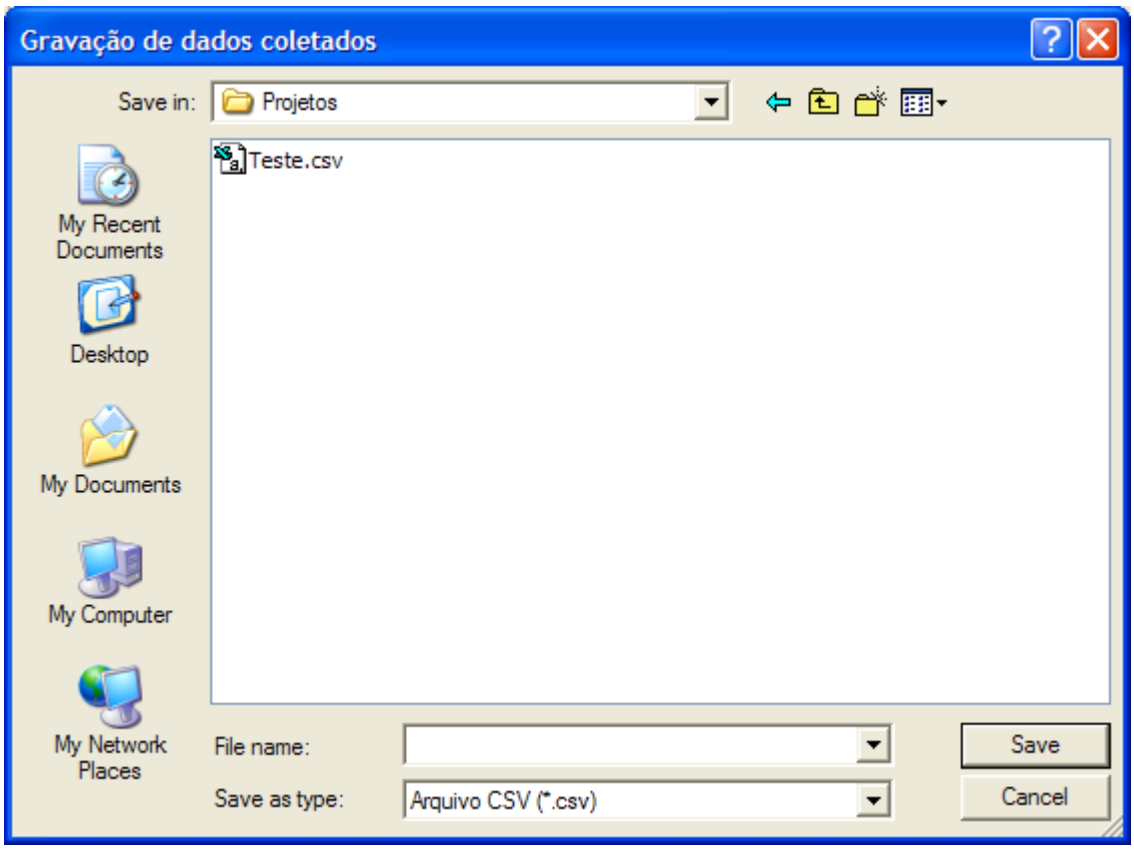
Nova Janela de Monitoração...

Esta opção cria uma nova janela de monitoração gráfica. Com isso, é possível monitorar graficamente variáveis do Controlador µDX100, e salvar os valores lidos em arquivos compatíveis com planilhas eletrônicas como Excel. Inicialmente é criada uma janela com as seguintes possibilidades:

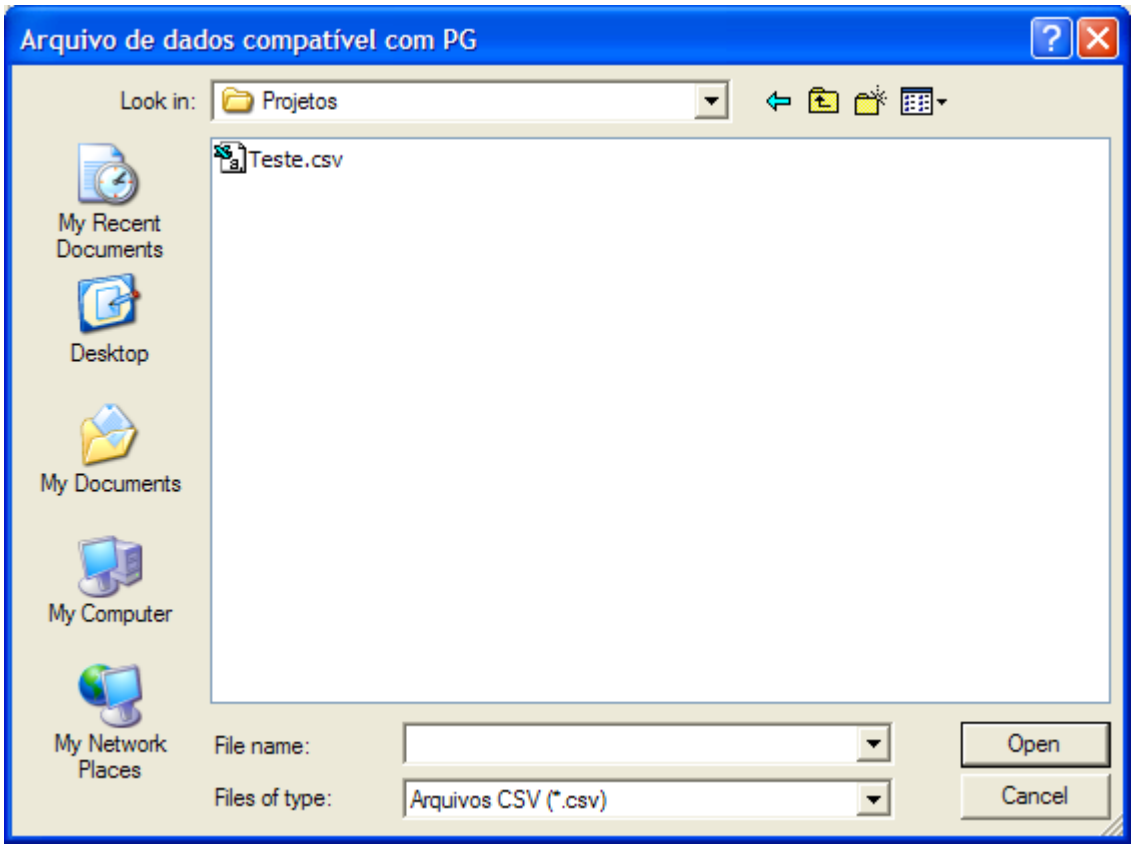


O **Modo de Aquisição** pode ser de três tipos:

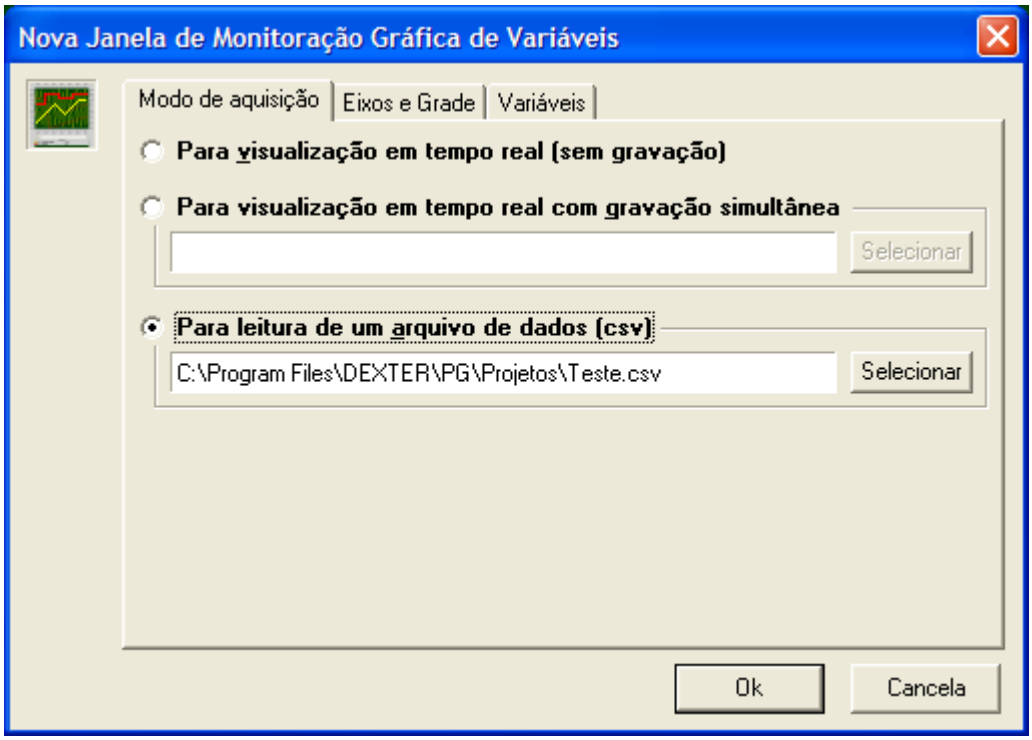
- **Para visualização em tempo real (sem gravação):** neste caso os dados irão aparecer na tela em tempo real, mas não serão gravados. Evidentemente, neste caso está descartado poder navegar para áreas anteriores do gráfico (ao completar a tela disponível todo o gráfico passa a rodar horizontalmente, descartando os dados mais antigos).
- **Para visualização em tempo real com gravação simultânea:** idêntico ao caso anterior, mas os dados lidos são também gravados em arquivo de tipo CSV, passível de ser lido posteriormente tanto no próprio PG como em uma planilha eletrônica como Excel, por exemplo. Ao selecionar esta opção surge uma janela para seleção do nome do arquivo a ser gerado com os dados:



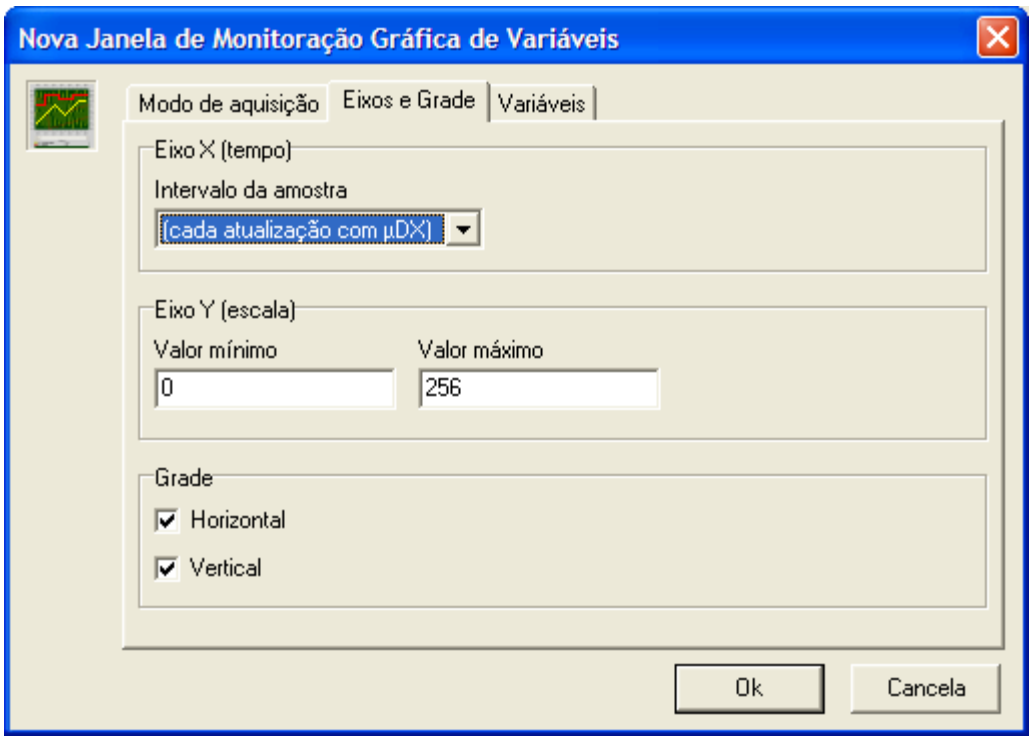
- **Para leitura de um arquivo de dados (CSV):** permite ler um arquivo do tipo CSV gravado anteriormente via opção anterior com dados do µDX100. Neste caso surge uma janela para seleção do arquivo a ser lido:



Ao selecionar uma das opções acima duas outras abas são ativadas na janela **Nova janela de Monitoração Gráfica de Variáveis**. São elas **Eixos e Grade** e **Variáveis**.

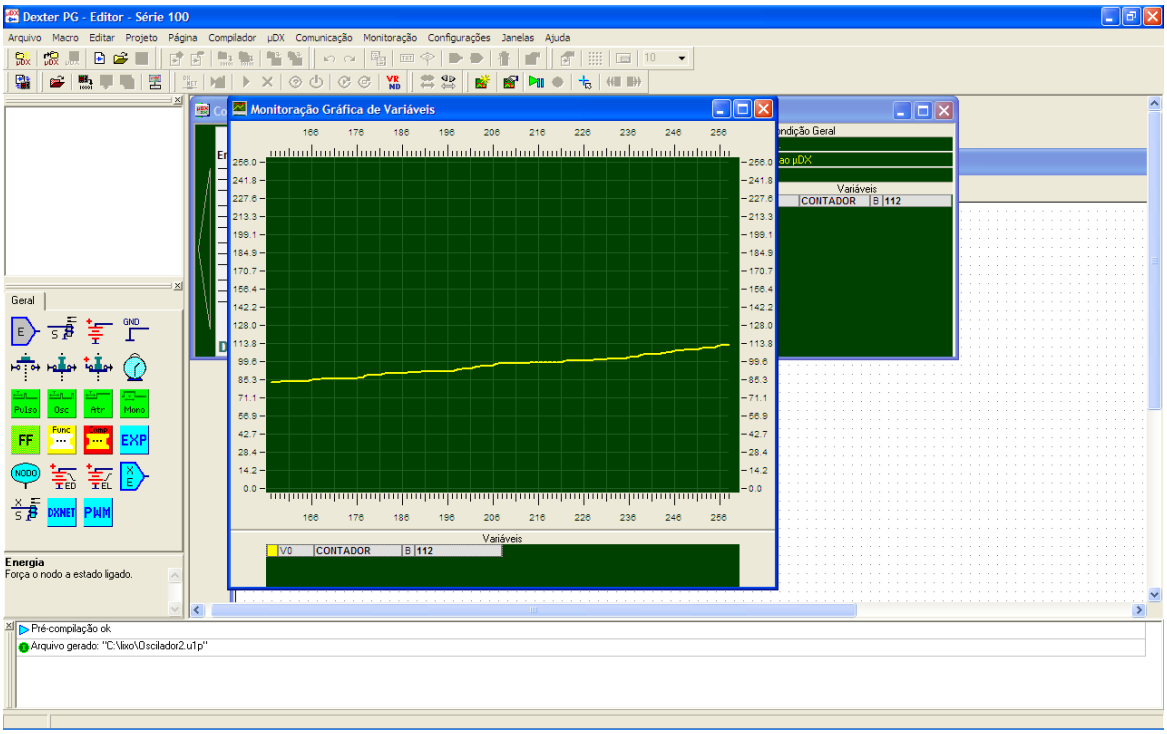


Na aba **Eixos e Grade** é possível selecionar o valor mínimo e máximo do eixo Y do gráfico (o valor default é 0 e 256), o intervalo de amostra do eixo X (o default é a cada atualização com µDX, ou seja, a velocidade máxima de monitoramento do PG). Por fim, também pode-se inibir a grade horizontal ou vertical do gráfico.

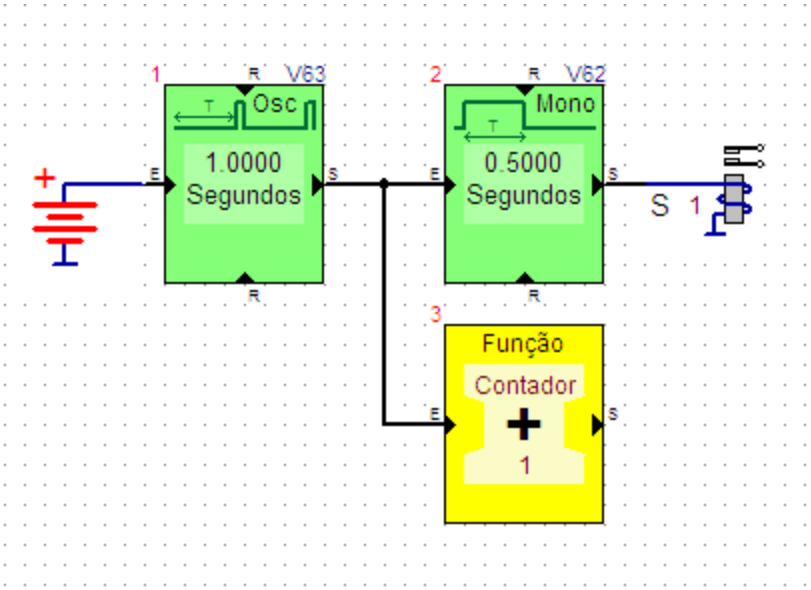


Na aba **Variáveis** seleciona-se quais variáveis serão plotadas no gráfico. Para que apareçam variáveis do µDX nas possibilidades de seleção desta janela é preciso que o Compilador PG esteja com a monitoração ativada (se não somente data e hora do microcomputador PC estarão disponíveis). Para selecionar as variáveis a serem plotadas basta clicar duas vezes sobre as mesmas, ou selecioná-las e clicar no botão  . O botão  permite eliminar variáveis da plotagem. O botão  possibilita editar o nome da variável e sua cor.

Caso o programa aplicativo existente no µDX100 coincida com o programa aplicativo carregado no Compilador PG (verificável via opção **Verifica programa no µDX**) estarão disponíveis para plotagem todas as variáveis usadas no programa do CLP. Ao clicar em Ok é gerada a janela de monitoração gráfica. A janela de monitoração gráfica a seguir foi gerada com captura da variável CONTADOR, que é incrementada a cada segundo:



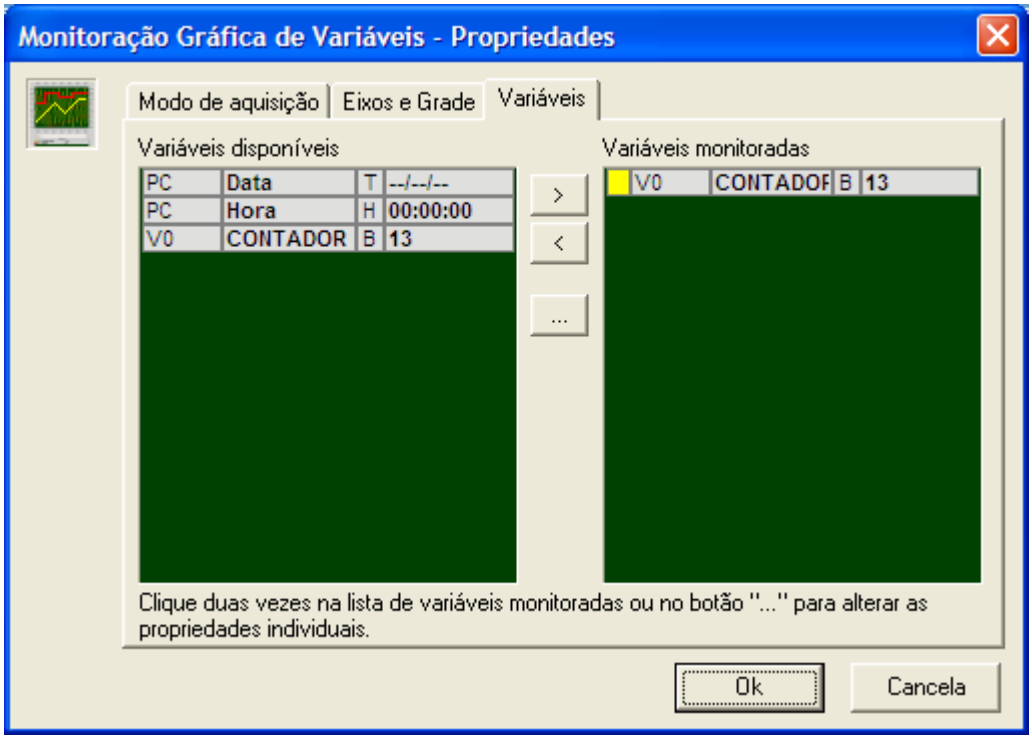
O programa aplicativo que gerou esta variável é:



Exemplo de Programa Aplicativo: **Oscilador2.d1g** (fonte); **Oscilador2.u1p** (compilado).

Propriedades

Acessa os parâmetros do gráfico, como escala X e Y, variáveis monitoradas, etc., permitindo modificá-los.

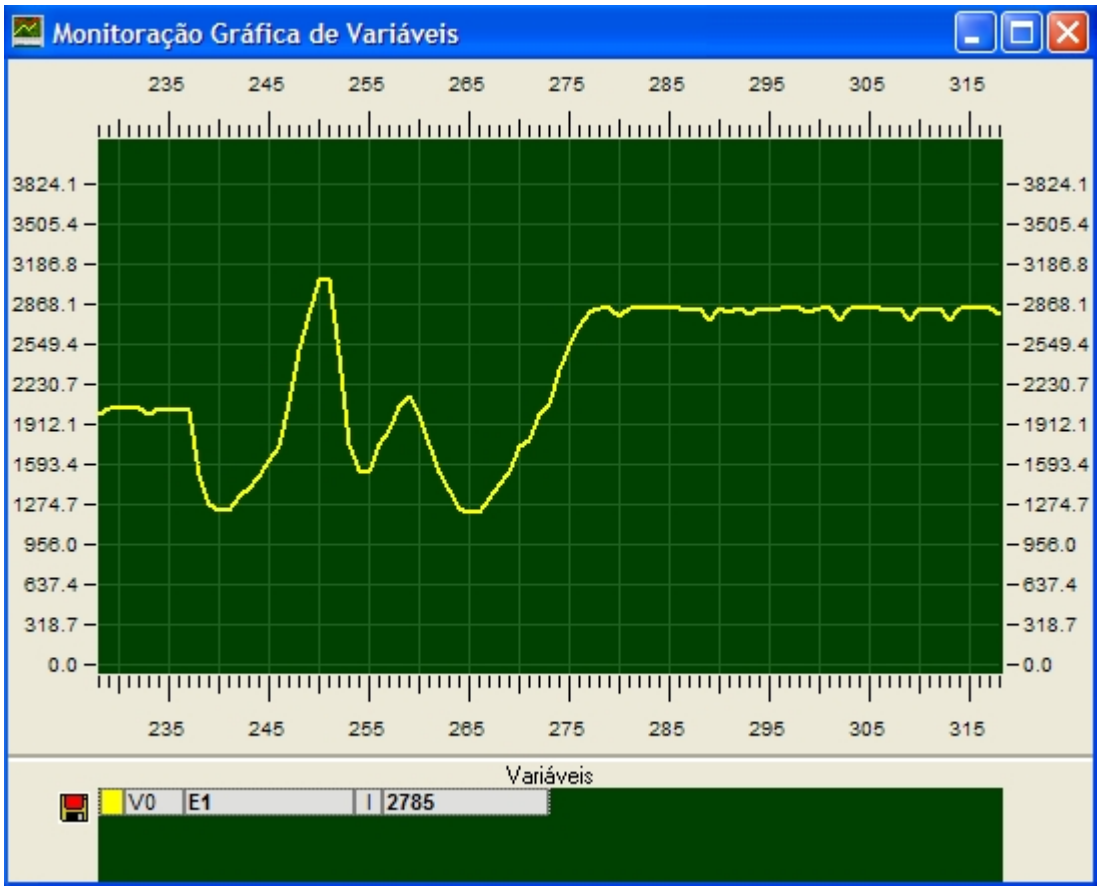


Lê Dados/Pausa

Interrompe ou reinicia a leitura de dados do gráfico.

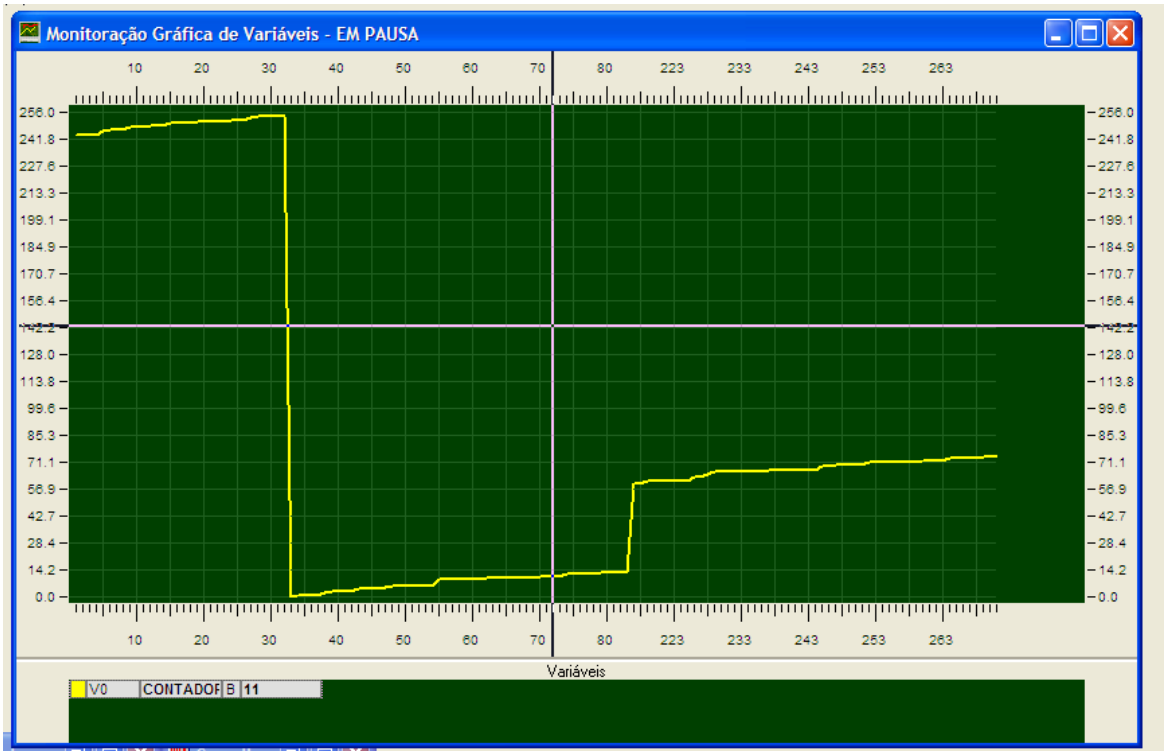
Grava

Interrompe ou reinicia a gravação dos dados lidos em arquivo. Note que a gravação dos dados é sinalizada por um símbolo de diskette em vermelho no canto inferior esquerdo da janela de monitoração.



Cursor

Cria um cursor sobre o gráfico, permitindo a leitura dos valores das variáveis em qualquer ponto do mesmo.



Retrocede Página (PgUp)

No caso de leitura de arquivo de dados (CSV) é permitido avançar e retroceder o gráfico. Esta opção retrocede em uma página completa o gráfico. Pode ser usada a tecla PgUp do teclado do computador. Note que o arquivo CSV pode ser bastante extenso (gerado via janela de monitoração gráfica com gravaçào ativada).

Avança Página (PgDn)

Esta opção avança em uma página completa o gráfico. Pode ser usada a tecla PgDn do teclado.

Retrocede (Left)

Retrocede em um registro o gráfico. Note que pode ser usada a seta para esquerda do teclado (Left).

Avança (Right)

Avança em um registro o gráfico. Note que pode ser usada a seta para direita do teclado (Right).

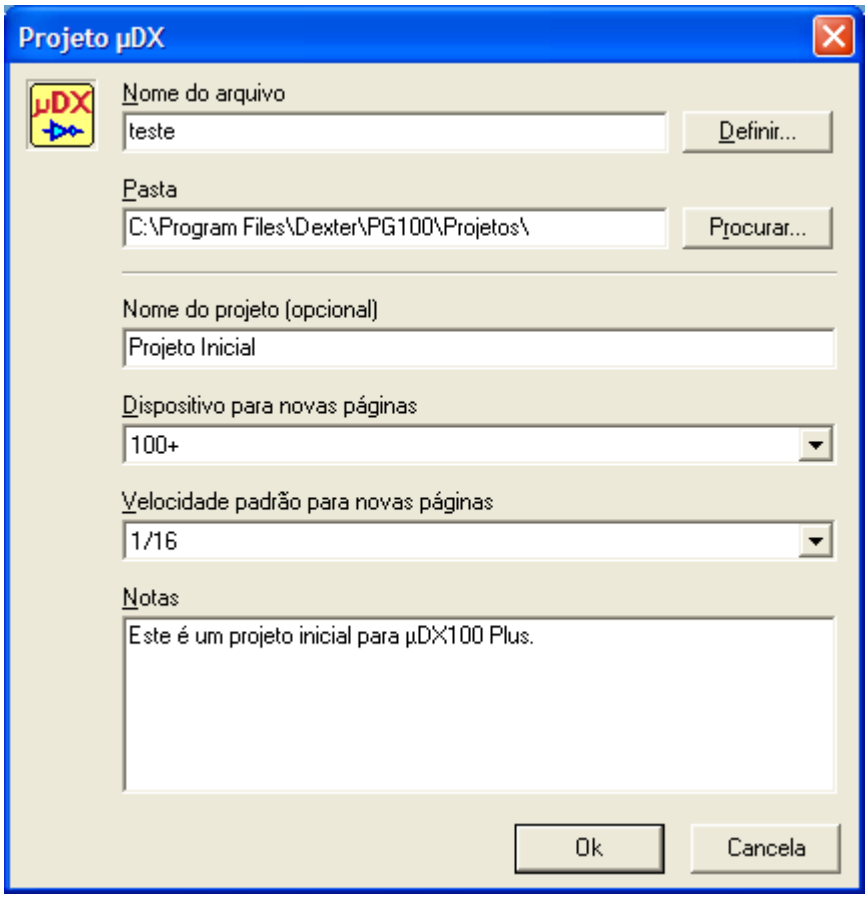
Parte VII

Elaborando Programas

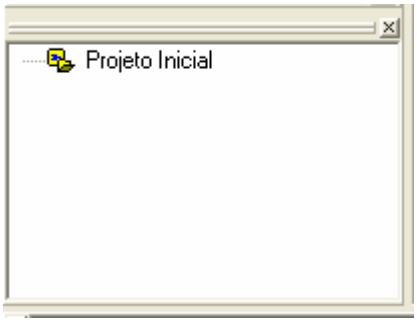
A criação de um programa para o µDX100 é muito fácil e pode ser dividida em quatro etapas:

- Abertura de um projeto para abrigar todas as páginas do programa.
- Confecção das páginas de programação.
- Pré-compilação do projeto.
- Compilação e envio do programa para o µDX100.

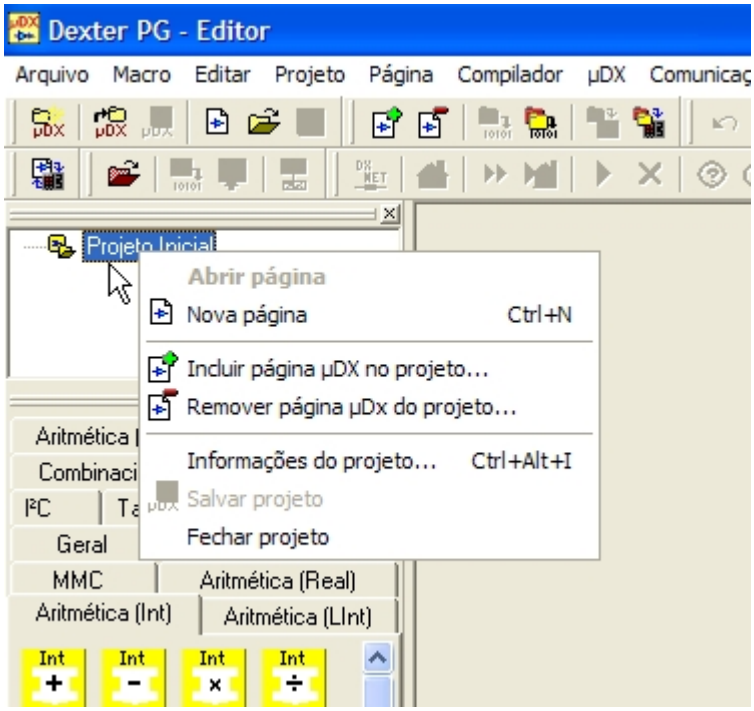
Então, acione o software Editor PG e pressione a tecla  existente na Barra de Ferramentas para gerar um novo projeto. Ou então abra o menu pop-down [**Arquivo**] → [**Novo projeto...**]. Irá surgir uma janela com campos de informação a serem preenchidos sobre o novo projeto. Digite as informações abaixo e pressione a tecla Ok.



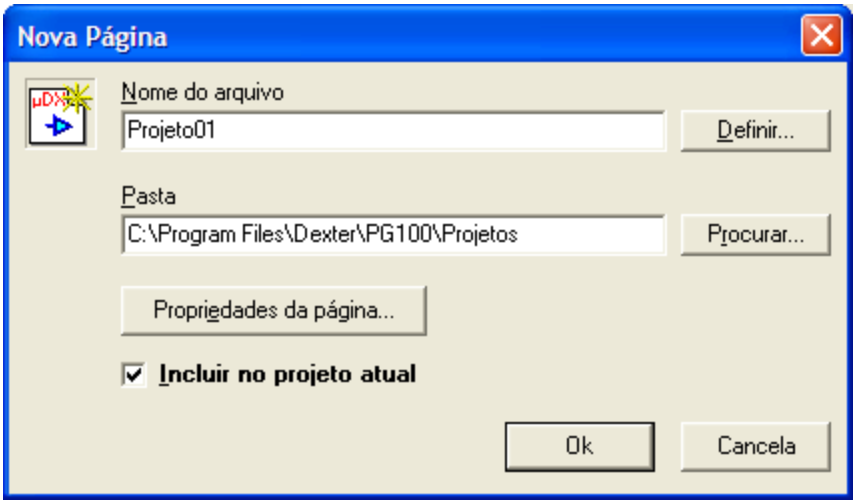
Surgirá uma pequena janela com o fontes do projeto criado (no caso, o projeto ainda não possui nenhuma página de programação):



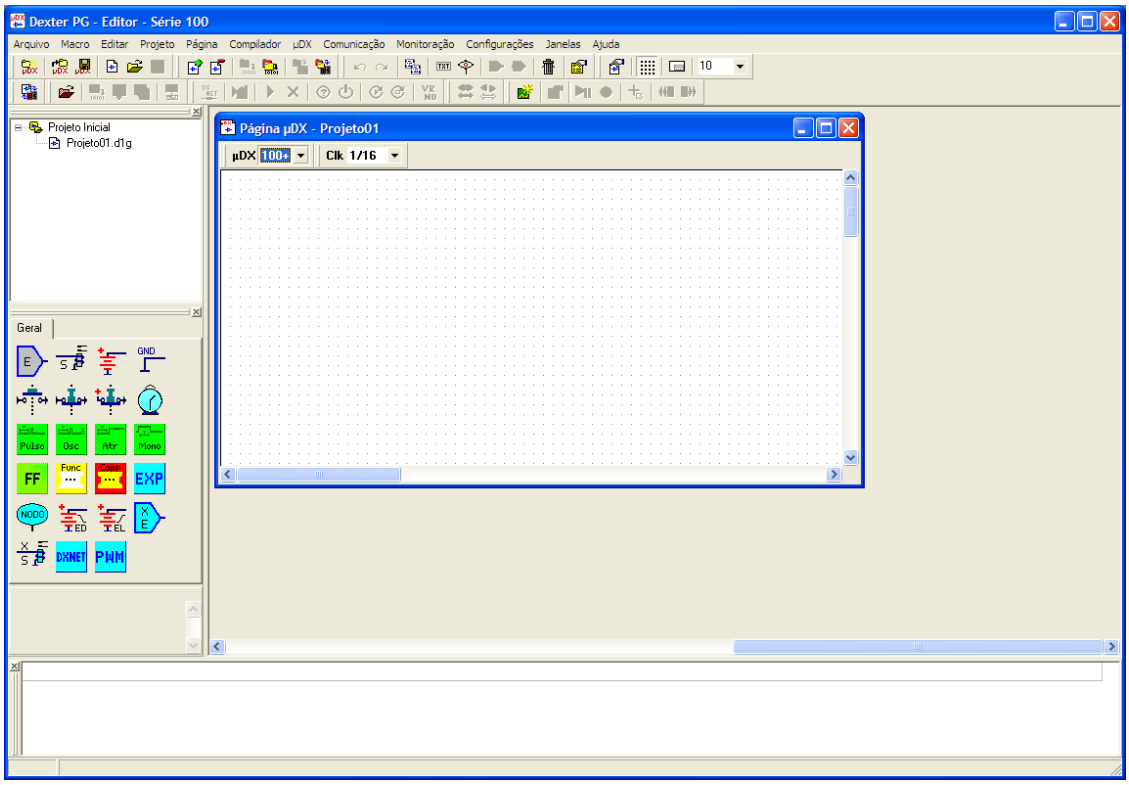
Pressionando a tecla direita do mouse sobre o projeto surgem várias opções, como inserção de páginas, visualização de informações sobre o projeto, etc.



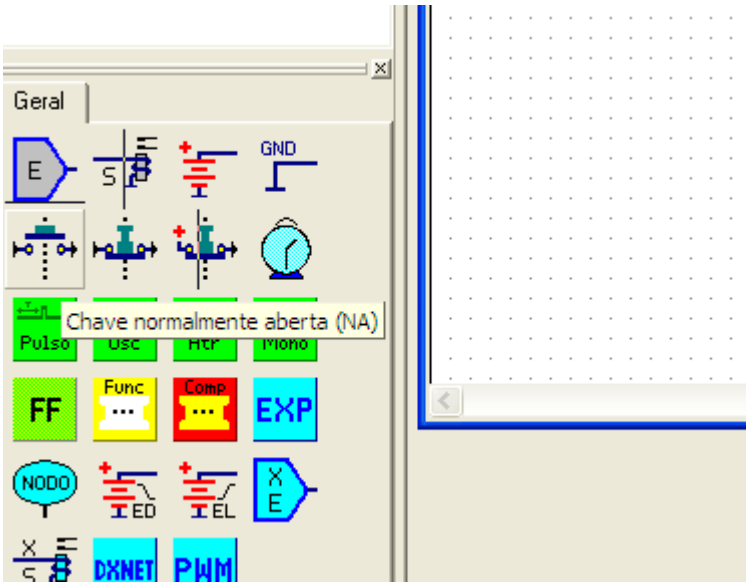
Clique em [Nova Página] para abrir uma página de programação vinculada ao projeto (ou utilize a tecla  existente na Barra de Ferramentas). Irá surgir uma janela com informações sobre a página. Preencha como abaixo:



Deverá surgir uma página de programação, chamada "Projeto01", na tela do Editor PG. Note que esta página está vinculada ao projeto chamado "Projeto Inicial". Caso não se queira gerar um projeto pode-se criar uma página avulsa apenas pressionando a tecla  existente na Barra de Ferramentas, sem a necessidade de primeiro gerar um projeto. O aspecto da tela do PG deve ser o seguinte:

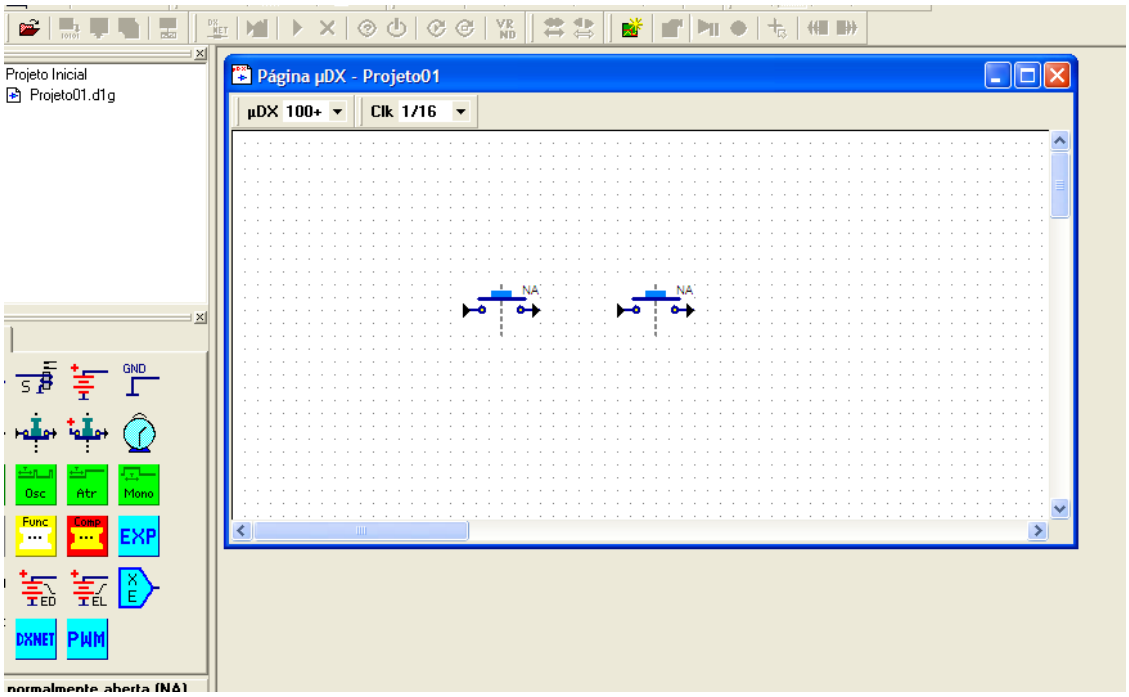


Podemos a partir deste ponto iniciar o desenvolvimento de um programa aplicativo. Digamos um programa bem simples, que ligue a saída S1 do µDX100 sempre que as entradas E1 e E2 deste controlador forem energizadas simultaneamente. Para isso pode-se usar chaves NA (normalmente aberta). Pressione a tecla esquerda do mouse sobre o bloco Chave NA:

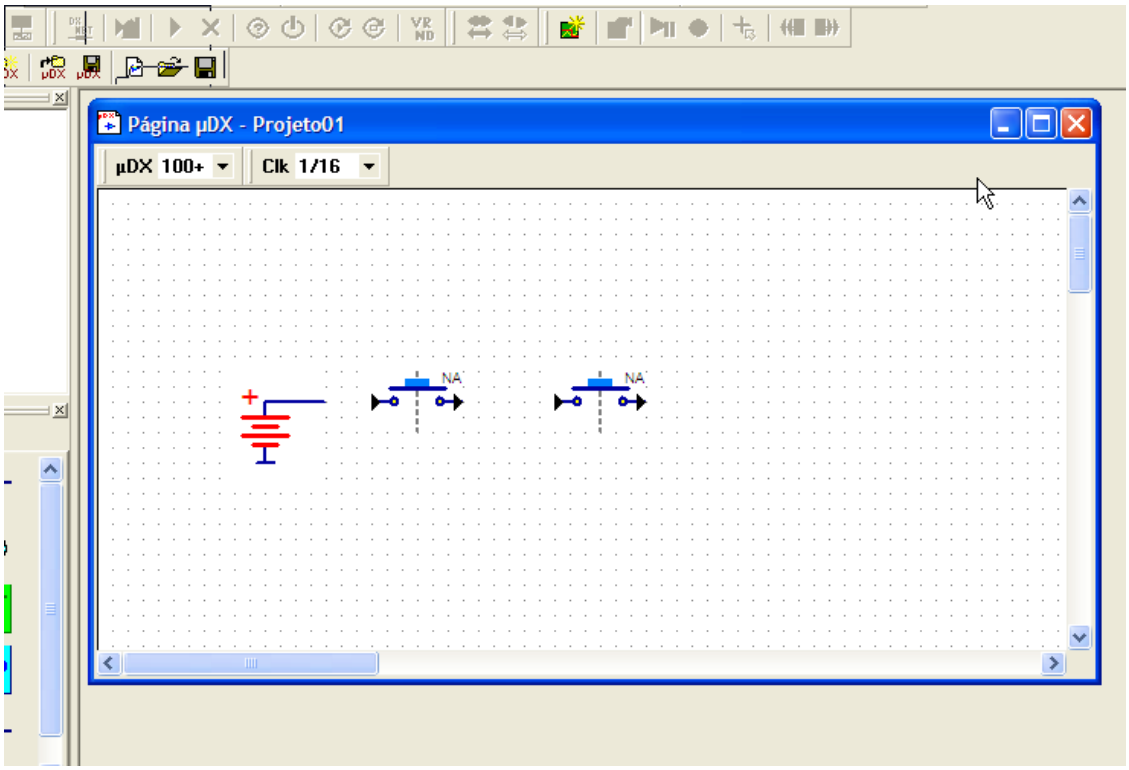


***Nota**: No software Editor PG deve-se pressionar a tecla esquerda do mouse uma vez sobre o bloco para capturá-lo e clicar novamente para largá-lo na janela de programação. Não arraste o bloco com a tecla esquerda do mouse pressionada.*

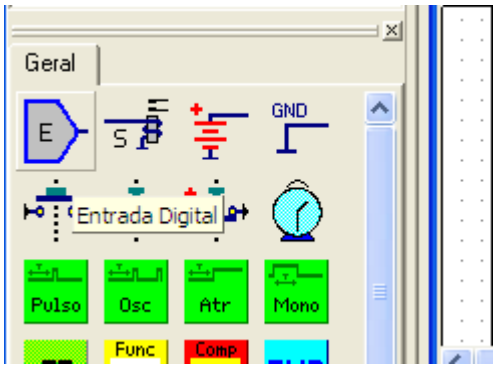
Monte desta forma a disposição mostrada abaixo:



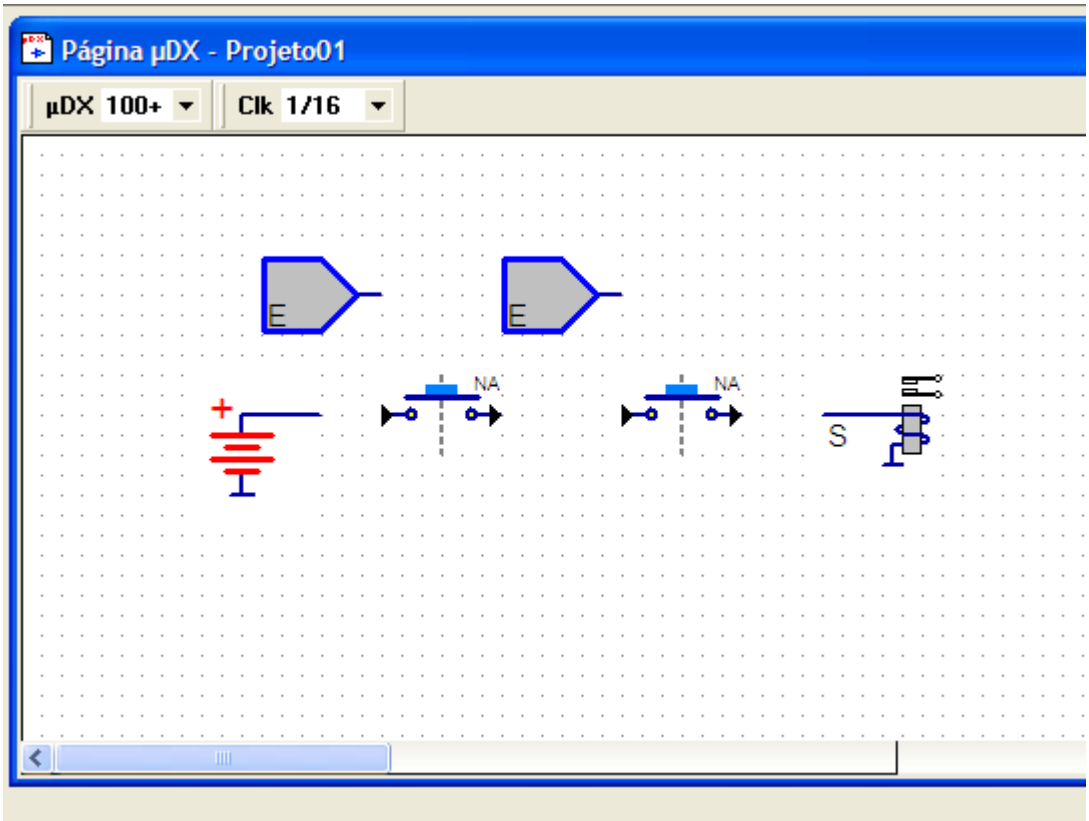
A seguir, coloque um bloco de Energia:



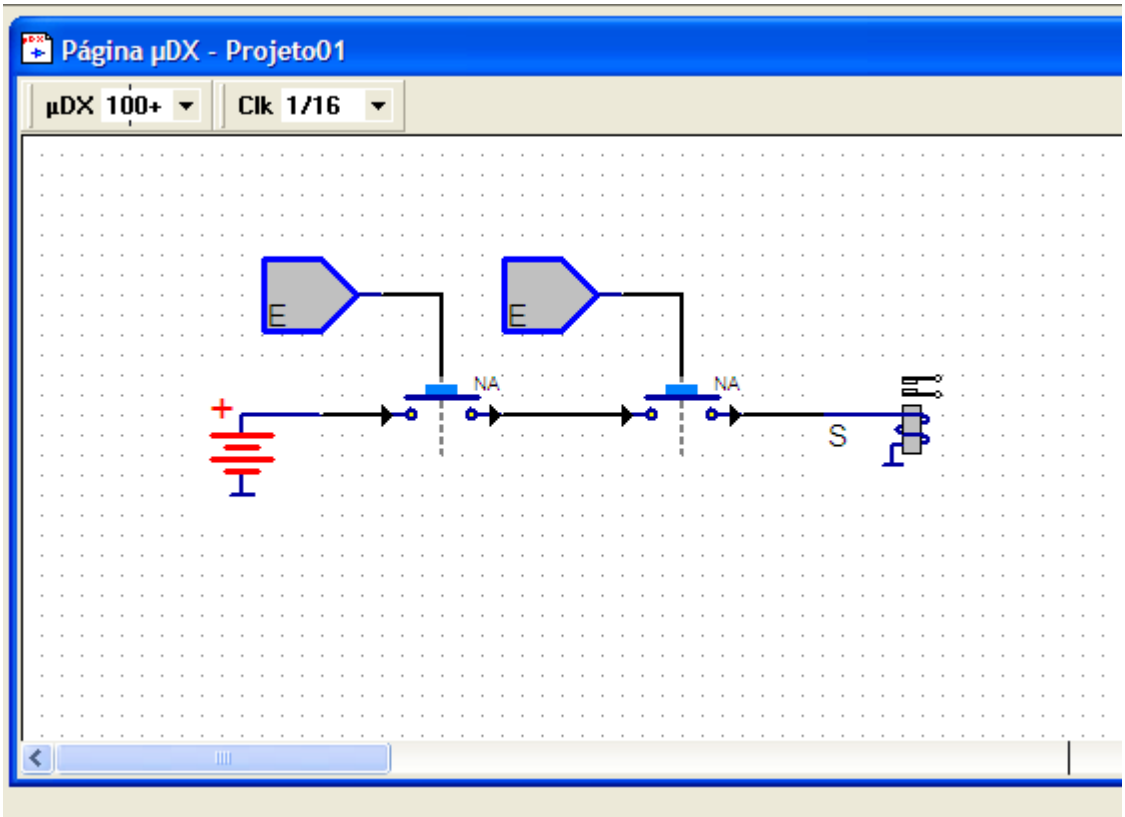
Por fim, vamos colocar os blocos de Entrada do controlador:



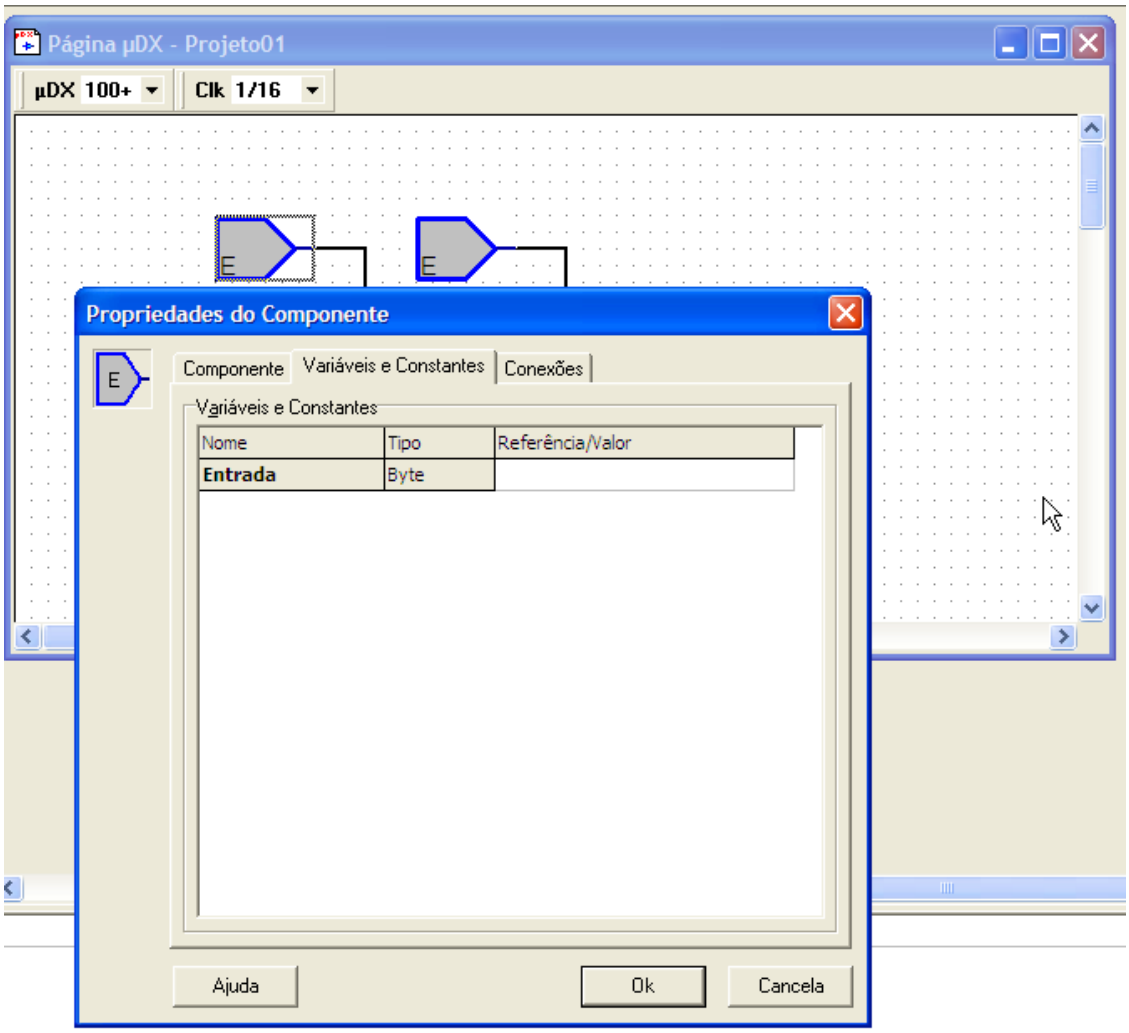
O aspecto final da disposição dos blocos na página de programação será:



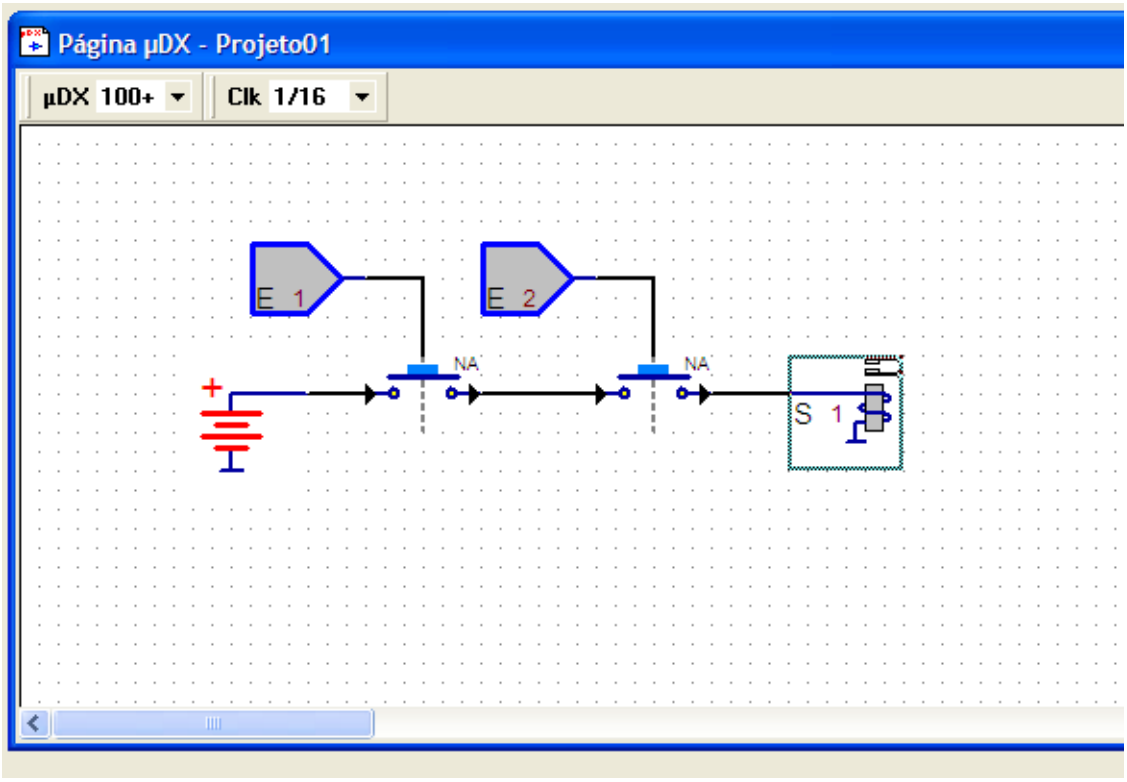
Agora basta conectar os blocos entre si e editar os blocos que possuem parâmetros. Para efetuar as conexões basta clicar com a tecla esquerda do mouse sobre as extremidades conectáveis dos blocos. Após todas as conexões o aspecto do programa é o seguinte:



Agora devemos editar os parâmetros dos blocos, no caso, os blocos de Entradas e Saídas Digitais. Para isso aponte para o bloco e pressione a tecla direita do mouse:



Edite os parâmetros destes blocos, selecionando Entrada 1 e 2, e por fim Saída 1:



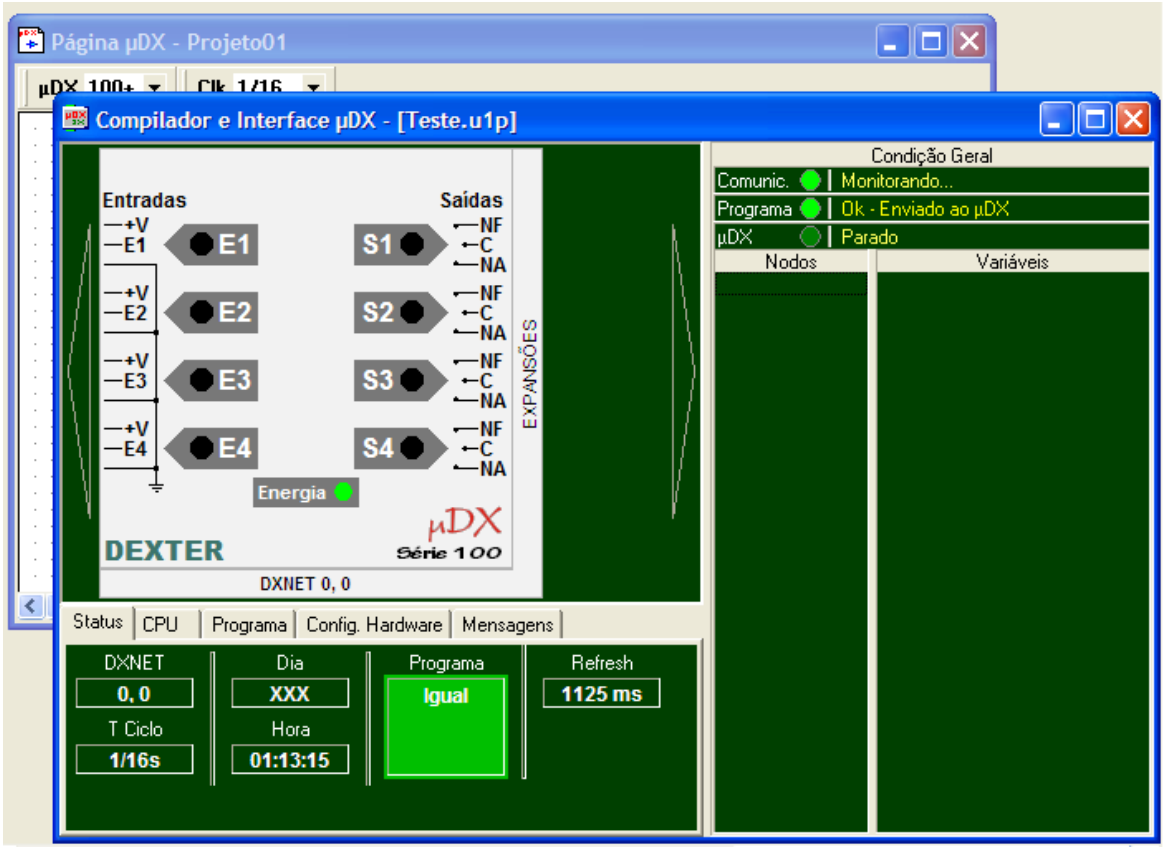
Exemplo de Projeto no PG: **Teste.d1p** (projeto); **Teste.u1p** (compilado).

Está pronto o programa! Note que a Energia só alcançará a Saída S1 se as duas entradas, E1 e E2, forem energizadas simultaneamente (fechando as chaves NA).

Agora devemos pré-compilar o programa no Editor PG, gerando o arquivo sufixo U1P que será lido no Compilador PG. Para isso clique no botão (compila projeto) existente na barra de ferramentas para compilar o projeto, ou pressione a tecla **F9** do computador, ou ainda utilize o menu pop-down [**Projeto**] → [**Compilar Projeto (compilador)**]. Deve ser ativado automaticamente o Compilador PG. Para comutar entre Editor PG e Compilador PG pressione a tecla (compilador/fontes) existente na barra de ferramentas do PG, ou a tecla **F5** do computador.

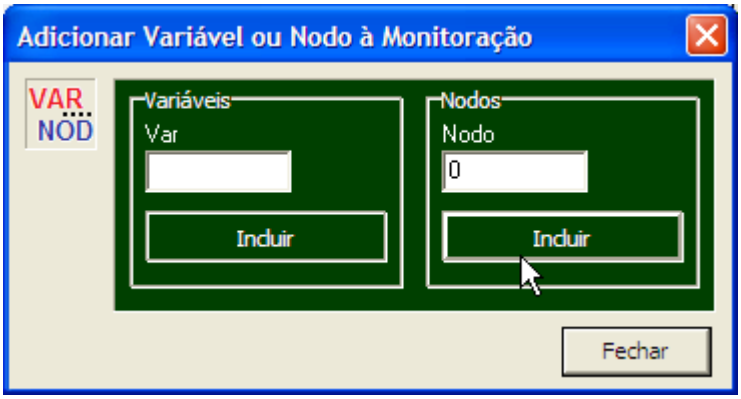
Irá surgir o Compilador PG, conforme mostrado na figura a seguir. Caso o µDX100 já esteja conectado a porta serial do computador correta o Compilador PG irá mostrar os dados de status do µDX100. Caso contrário conecte o µDX100 à porta serial do computador via Modem para µDX100 (ou a um cabo de adaptação USB - RS232) e siga os passos descritos no capítulo Teclas de Operação do Compilador.

Se a comunicação foi estabelecida com o µDX100, podemos carregar o programa no controlador µDX100. Para isso pressione a tecla (enviar programa) existente no PG.

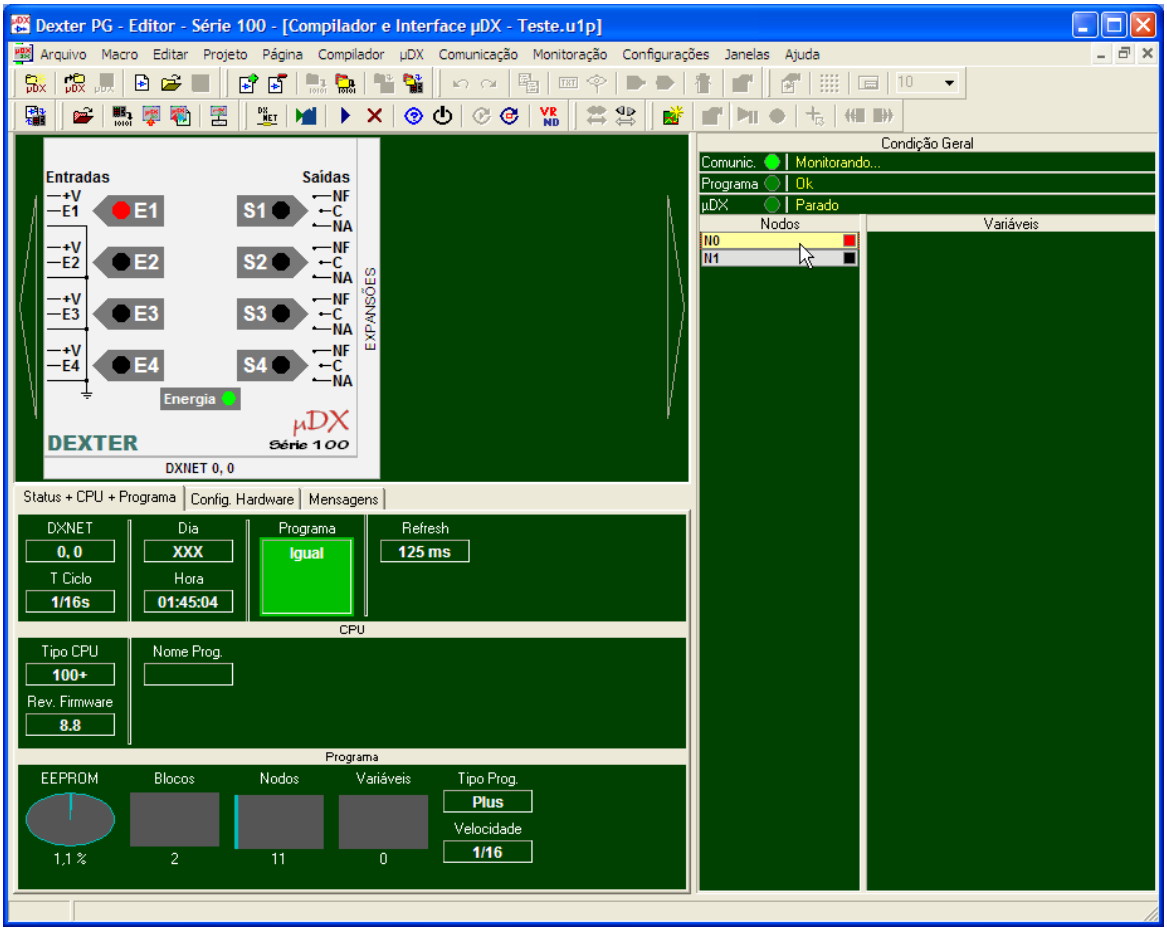


Por fim, pressione a tecla  para executar o programa transmitido. A tela do Compilador PG deve apresentar o led de Exec ligado, indicando que o programa aplicativo está sendo executado no µDX100.

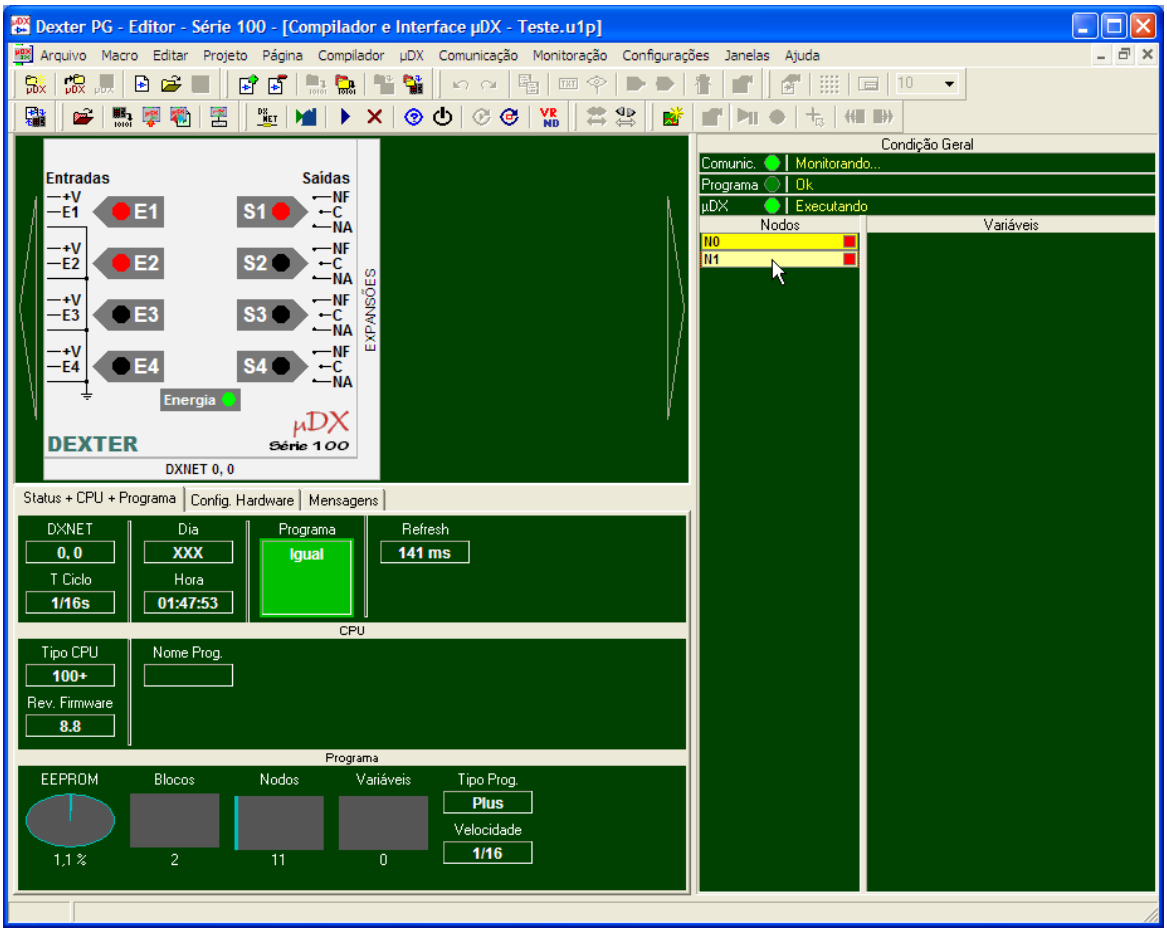
Agora é possível testar o programa. Para isso devemos incluir na lista de nodos monitorados os nodos 0 e 1 (entradas E1 e E2 do controlador µDX100, respectivamente). Pressione a tecla  ou selecione a opção **Adicionar variáveis ou nodos à monitoração...** existente no menu pop-down **Compilador**. Adicione os nodos 0 e 1:



Com isso estes nodos surgem na lista de nodos monitorados. Se for pressionado duas vezes sobre o nodo na lista podemos força-lo a ligar. Para indicar que o nodo está sendo forçado em 1 pelo PG ele é representado em amarelo. Note que o led que representa a entrada E1 no desenho do µDX100 liga, já que o nodo 0 está associado a esta entrada:



Se fizermos o mesmo com o nodo 1 (associado à entrada E2 do µDX100) deverá ser acionada a saída S1 do µDX100, pois as duas chaves NA estarão fechadas, permitindo a energização desta saída. Ou seja, fica bastante simples depurar o programa. Óbvio que é possível também energizar efetivamente as entradas do controlador µDX100 para testar o programa.


Dexter PG - Editor - Série 100 - [Compilador e Interface µDX - Teste.u1p]
Arquivo Macro Editar Projeto Página Compilador µDX Comunicação Monitoração Configurações Janelas Ajuda
Entradas
Saídas
E1 E2 E3 E4
S1 S2 S3 S4
NF C NA
EXPANSÕES
Energia
DEXTER
µDX
Série 100
DXNET 0, 0
Condição Geral
Comunic. Monitorando...
Programa Ok
µDX Executando
Nodos
Variáveis
N0
N1
Status + CPU + Programa
Config. Hardware
Mensagens
DXNET
0, 0
Dia
XXX
Programa
Igual
Refresh
141 ms
T Ciclo
1/16s
Hora
01:47:53
CPU
Tipo CPU
100+
Nome Prog.
Rev. Firmware
8.8
Programa
EEPROM
Blocos
Nodos
Variáveis
Tipo Prog.
Plus
Velocidade
1/16
1,1 %
2
11
0

Convenções do Compilador PG

O Compilador PG utiliza as seguintes convenções para acesso a memória, nodos e variáveis do controlador µDX100:

Vnnnn → Acessa variável absoluta do controlador.

Nnnnn → Acessa nodo absoluto do controlador.

Para utilizar números hexadecimais basta incluir o sufixo **h**. Caso o número inicie por uma letra é preciso incluir um zero à esquerda para que o PG não confunda o valor literal com o nome de alguma variável. Assim, por exemplo, o valor 245 em hexadecimal seria representado por **0F5h**.

O controlador µDX100 utiliza os seguintes tipos de variáveis:

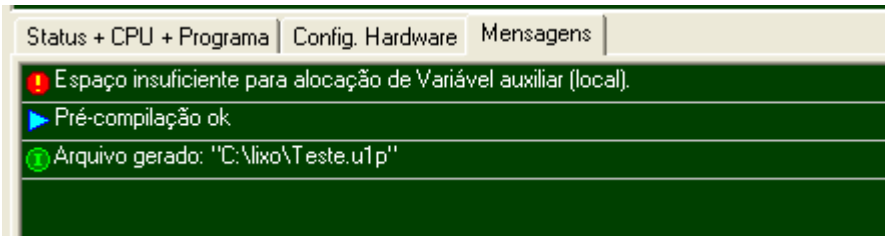
Nodo → Constantes e variáveis entre 0 e 1.

Byte → Apenas constantes entre 0 e 255.

Variável Absoluta (Vnnnn)

A letra V seguida de um valor numérico acessa uma variável absoluta do CLP. Note que o µDX100 possui 16 variáveis (de v0 a v15) e o µDX100+ possui 64 variáveis (de v0 a v63). Todos os blocos de temporizadores (atraso, monoestável, oscilador, pulso) utilizam uma variável auxiliar para fazer a contagem de tempo. Estas variáveis auxiliares são alocadas a partir da última variável disponível (a partir de v15 no caso de µDX100, ou de v63 no caso de µDX100+, ou ainda v255 no caso de µDX101). É preciso cuidado no uso de variáveis no programa do µDX100, pois a lista de variáveis auxiliares alocadas por estes blocos de temporizadores não deve se sobrepor as variáveis usadas no programa aplicativo.

Quando isso ocorre é gerado um erro no Compilador:



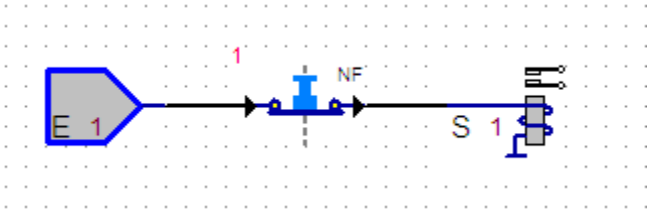
O uso de variáveis absolutas é necessário quando estas variáveis devem ser acessíveis a outro controlador µDX100 na rede DXNET, ou a um programa supervisório (já que estes dispositivos externos não sabem os nomes que foram atribuídos às variáveis do programa aplicativo, podendo endereçar apenas variáveis absolutas).

Nodo Absoluto (Nnnnn)

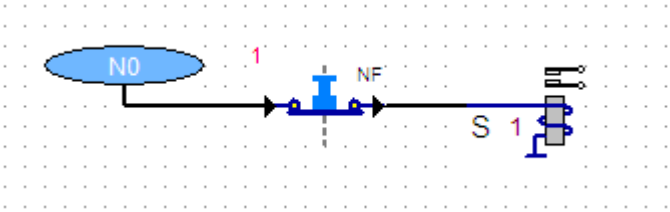
A letra N seguida de um valor numérico acessa um nodo absoluto do CLP. Novamente a maior utilidade de acesso a um determinado nodo específico é para disponibilizá-lo para um dispositivo externo (seja um outro µDX100 ligado a rede DXNET, seja um sistema supervisório ligado a porta serial). Os nodos N0 até N7 e os nodos N62 e N63 são reservados e possuem atribuições específicas. Ou seja, os nodos livres para uso no programa aplicativo iniciam em N8. No caso de µDX100 existem os nodos N0 a N63. Já no caso de µDX100+ os nodos vão de N0 a N191. Os nodos reservados são:

N0 **Entrada digital E1 do controlador µDX100.**
N1 **Entrada digital E2 do controlador µDX100.**
N2 **Entrada digital E3 do controlador µDX100.**
N3 **Entrada digital E4 do controlador µDX100..**
N4 **Saída digital S1 do controlador µDX100.**
N5 **Saída digital S2 do controlador µDX100.**
N6 **Saída digital S3 do controlador µDX100.**
N7 **Saída digital S4 do controlador µDX100.**
N62 **Sempre em zero.**
N63 **Sempre em um.**

Note que nodos são variáveis binárias, ou seja, assumem valor 1 ou 0 (verdadeiro ou falso). Todos estes nodos especiais estão disponíveis através de blocos específicos. Por exemplo, podemos ler o estado da entrada E1 tanto usando o bloco existente para este fim quanto acessando diretamente o nodo N0:

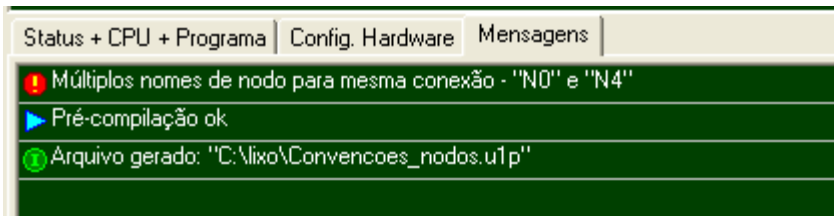


Exemplo de Programa Aplicativo: **Convencoes_nodo.d1g**



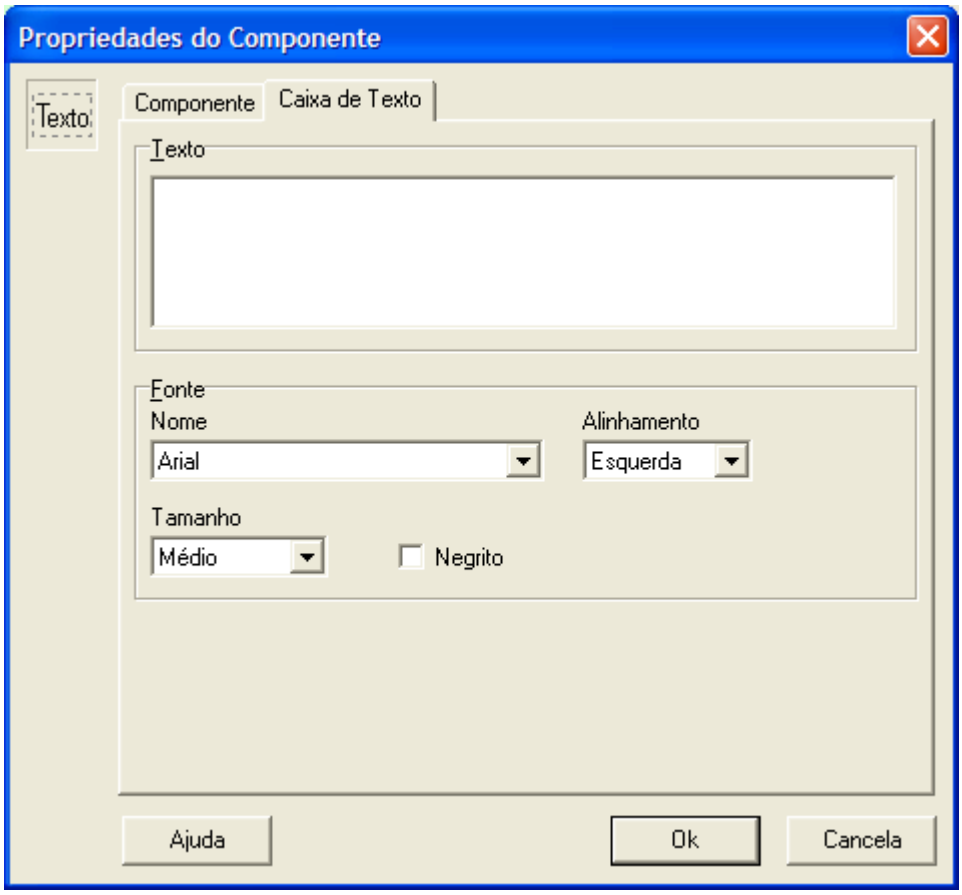
Exemplo de Programa Aplicativo: **Convencoes_nodo2.d1g**

Note que nos dois programas acima foi intercalada uma chave NF (normal fechada) entre a entrada E1 (ou nodo N0) e a saída S1. Isso é necessário porque as entradas e saídas do controlador µDX100 possuem números de nodos absolutos (N0 a N3 para entradas; N4 a N7 para saídas). Assim, não é possível conectar diretamente uma entrada a uma saída do controlador, pois iria atribuir dois nodos distintos a mesma conexão. Caso isso seja feito irá surgir o seguinte erro de compilação:



Texto

É possível inserir uma caixa de texto no programa. Esta caixa de texto não é utilizada pelo programa, servindo exclusivamente como comentário para o programa aplicativo do µDX100. Para "largar" a caixa de texto sobre o programa basta clicar com a tecla esquerda do mouse sobre o local desejado. Para editar a caixa de texto, permitindo inserir o texto, clique com a tecla direita do mouse apontando para a caixa de texto. Deve surgir uma janela que permite, além da digitação do texto, a escolha da fonte de caracteres a ser usada, o tamanho dos caracteres, e se em negrito ou não, além do alinhamento do texto.



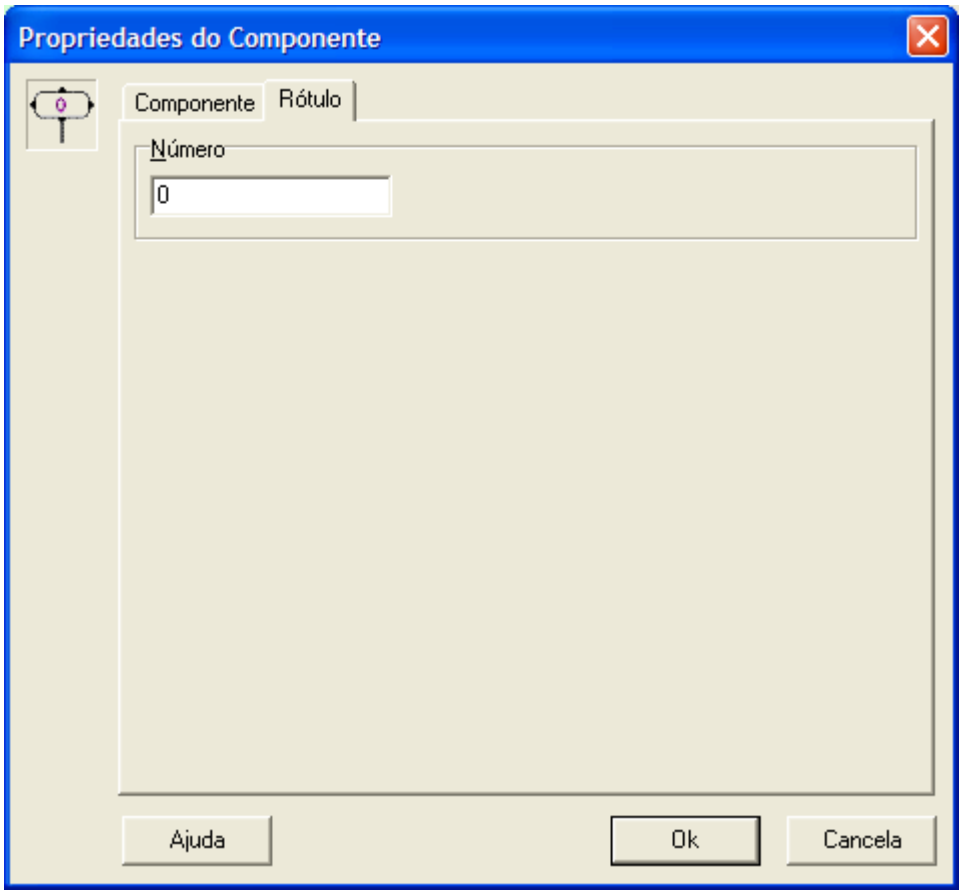
Para ajustar o tamanho da caixa de texto (de forma que o texto digitado caiba na caixa) basta selecionar a caixa e arrastar os cantos dessa, modificando suas dimensões.

Observação: É uma boa prática de programação incluir caixas de texto com comentários pertinentes sobre o funcionamento do programa aplicativo (tipo função das entradas e saídas, função de determinados blocos, etc.). Lembre-se que isso irá facilitar muito o entendimento do programa no futuro.

Rótulo

A função do rótulo é conectar diferentes pontos do programa entre si. O rótulo apenas conecta entre si todos os pontos com o mesmo número de rótulo (de 0 a 999), como se estivessem conectados por "fios", não atribuindo nenhum valor determinado ou nome para esta conexão. Ao editar o Rótulo surge a janela que permite atribuir um valor numérico ao mesmo. Todos os rótulos com mesmo valor numérico estarão conectados (mesmo em janelas de programação distintas de um mesmo projeto).

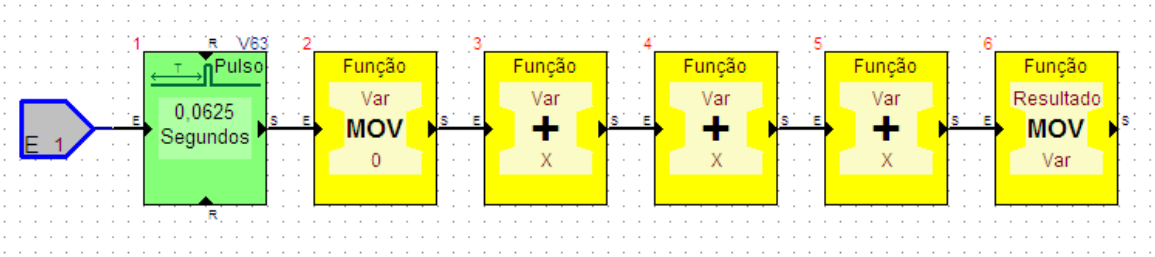
Note que rótulo não é permitido em Macros. Desta forma, esta função estará inibida quando estiver selecionada uma página de Macro.



Perceba a diferença sutil entre **Rótulo** e o bloco **Nodo**, existentes na aba **Geral** da Biblioteca de Componentes. O Rótulo apenas conecta dois pontos como se fossem utilizados "fios" de conexão. Já o bloco **Nodo** atribue um nome à conexão, ou atribui um nodo absoluto à conexão.

Macro

A Macro (macro-célula, ou macro-bloco) permite condensar todo um programa aplicativo em apenas um bloco, a ser utilizado em outros programas. Este recurso é muito poderoso, pois permite abordar uma determinada aplicação complexa a partir de blocos menores. Note que pode-se gerar uma nova Macro a partir de outras Macros pré-existentes. Evidentemente deve-se ter em mente que a Macro utiliza a mesma quantidade de blocos que a geraram, ou seja, não há economia de blocos no uso de Macros (apesar dela ser representada como um bloco único no programa aplicativo). Vamos exemplificar isso. Digamos que seja necessária a função matemática **f(x)=3x**. Esta função não está disponível nos blocos aritméticos do µDX100, mas existe a função f(x)=x+y. Então, a função desejada pode ser facilmente implementada:

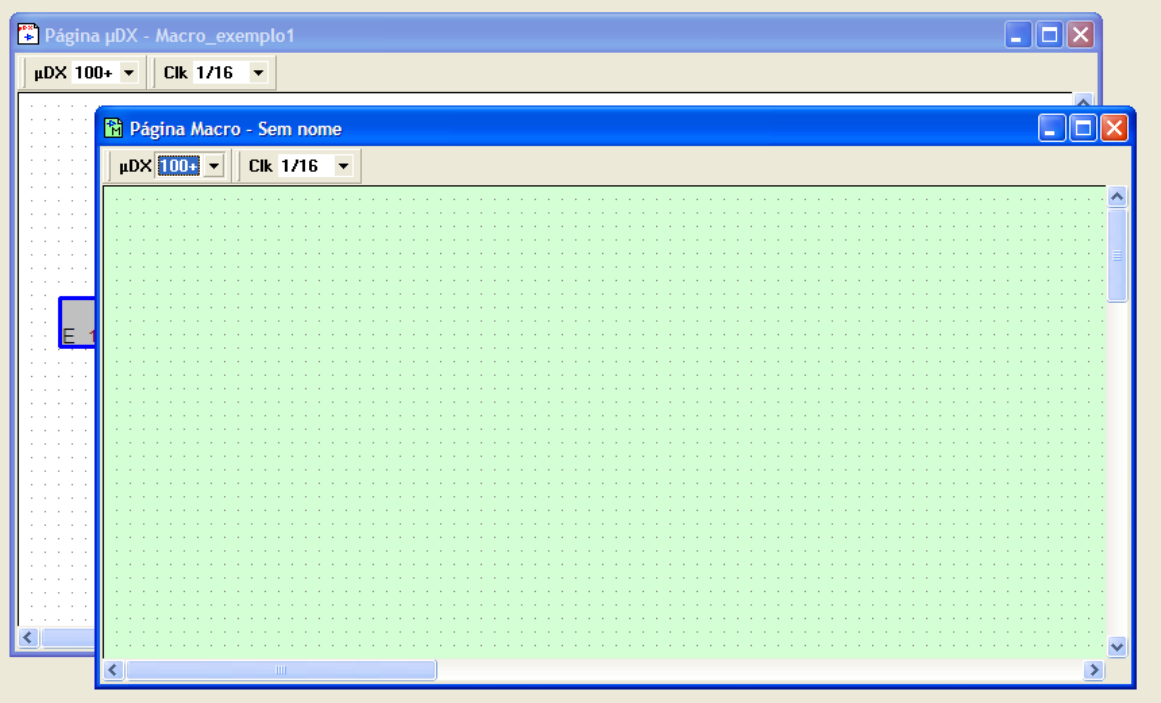


Exemplo de Programa Aplicativo: **Macro_exemplo1.d1g**

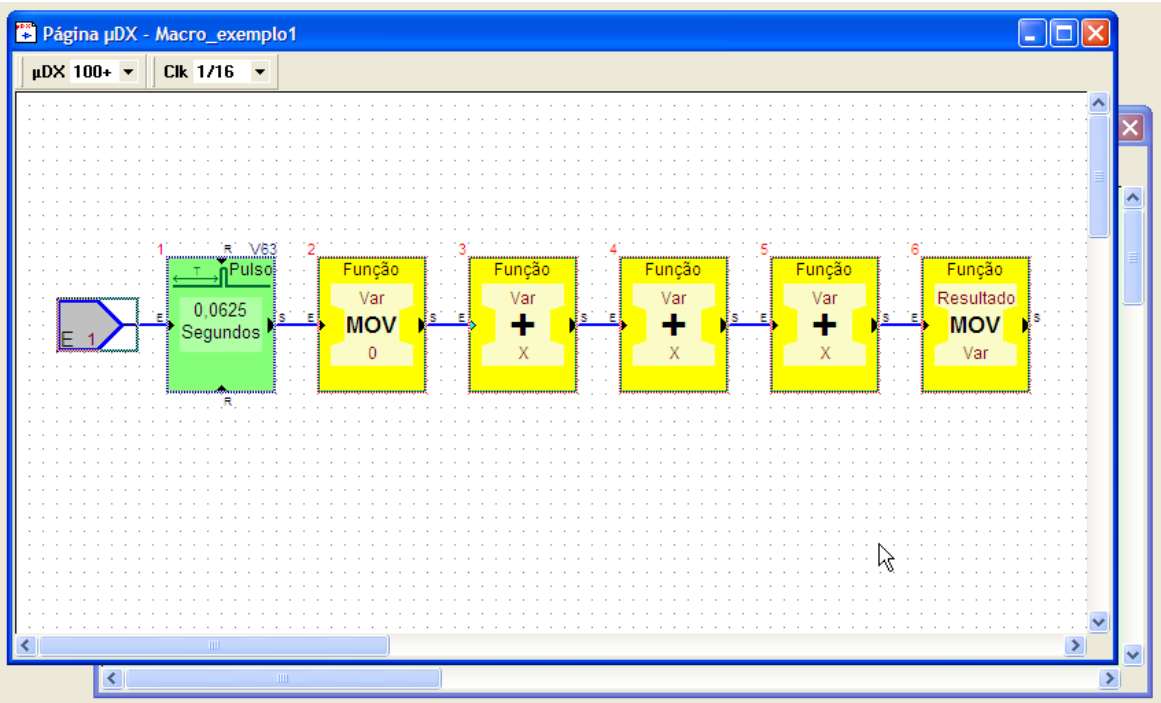
O programa aplicativo acima facilmente gera a função desejada. Note que a operação é iniciada ao energizar-se a entrada E1 do µDX100. Foi intercalado um bloco de Pulso para gerar apenas um pulso de um ciclo de duração, fazendo com que os blocos subseqüentes sejam executados apenas uma vez. Podemos testar seu funcionamento via Compilador PG. Por exemplo, se forçarmos a variável **x** com valor 4:

Note que a variável **Resultado** assumiu o valor 12 (4x3) ao energizar a entrada E1 do controlador µDX100.

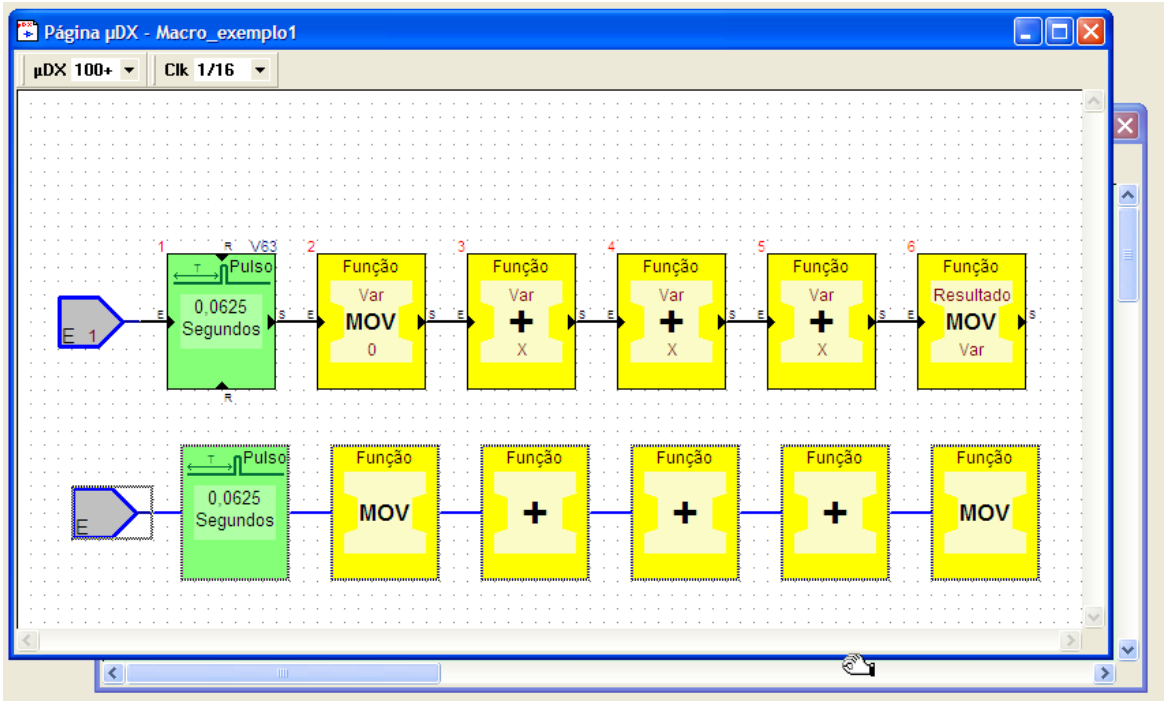
Vamos transformar este pequeno programa em uma Macro, capaz de ser utilizada em outros programas para o µDX100. Para isso vá em [**Arquivo**] → [**Nova Macro**]. Deve surgir uma nova janela (Página Macro). As janelas de programação de Macros possuem o fundo verde, para distinguí-las das janelas de programação do µDX, cujo fundo é branco.



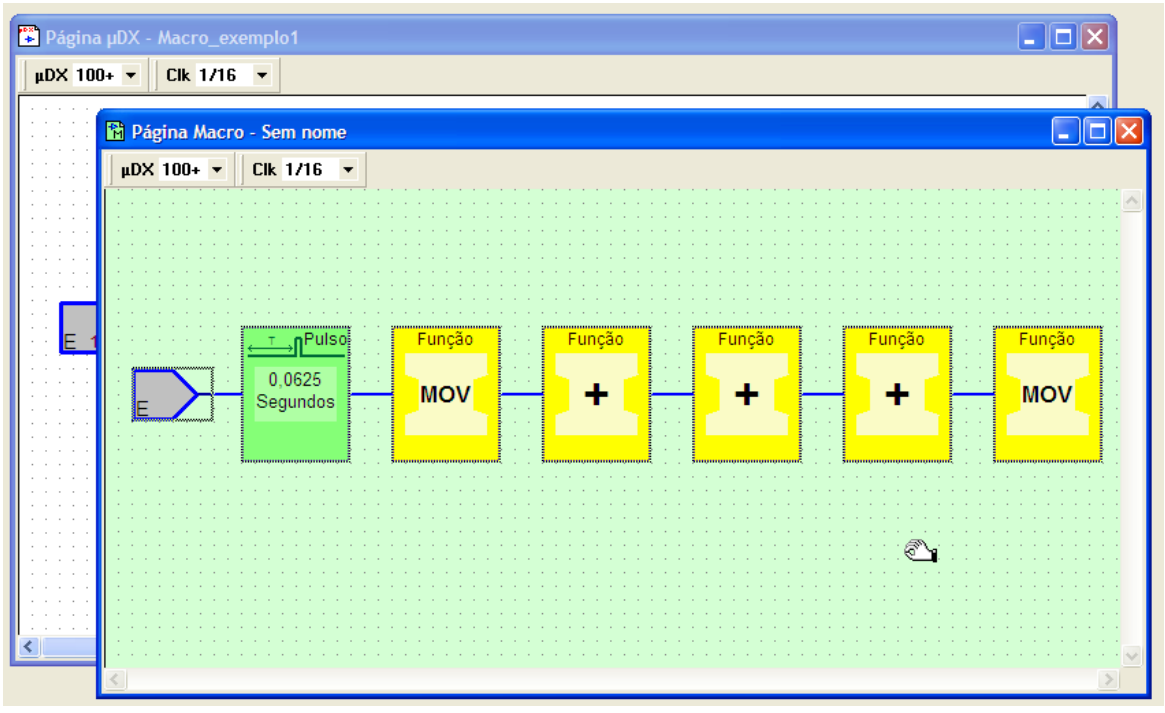
Retorne à Página µDX onde foi desenhado o programa e o selecione (para isso basta criar um retângulo de seleção mantendo pressionado o botão esquerdo do mouse e deslocar o cursor até abarcar todo o programa):



Agora pressione **Ctrl+C** no teclado do computador, ou vá no menu pop-down [**Editar**] → [**Duplicar**]. Com isso duplicamos o programa selecionado:

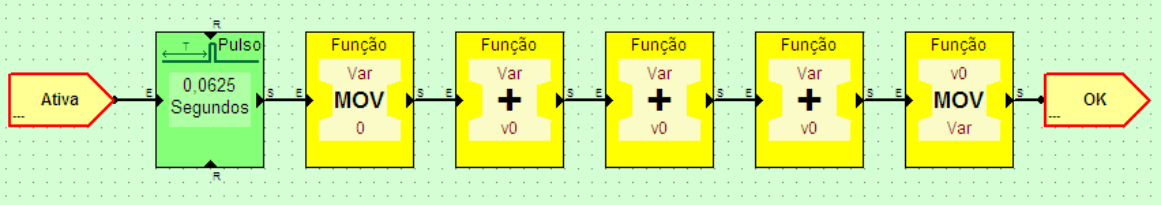


Agora clique com o botão esquerdo do mouse sobre a janela de programação de Macro (tela verde). Com isso o programa duplicado poderá ser "largado" nesta janela. Escolha um local adequado e pressione novamente o botão esquerdo do mouse:

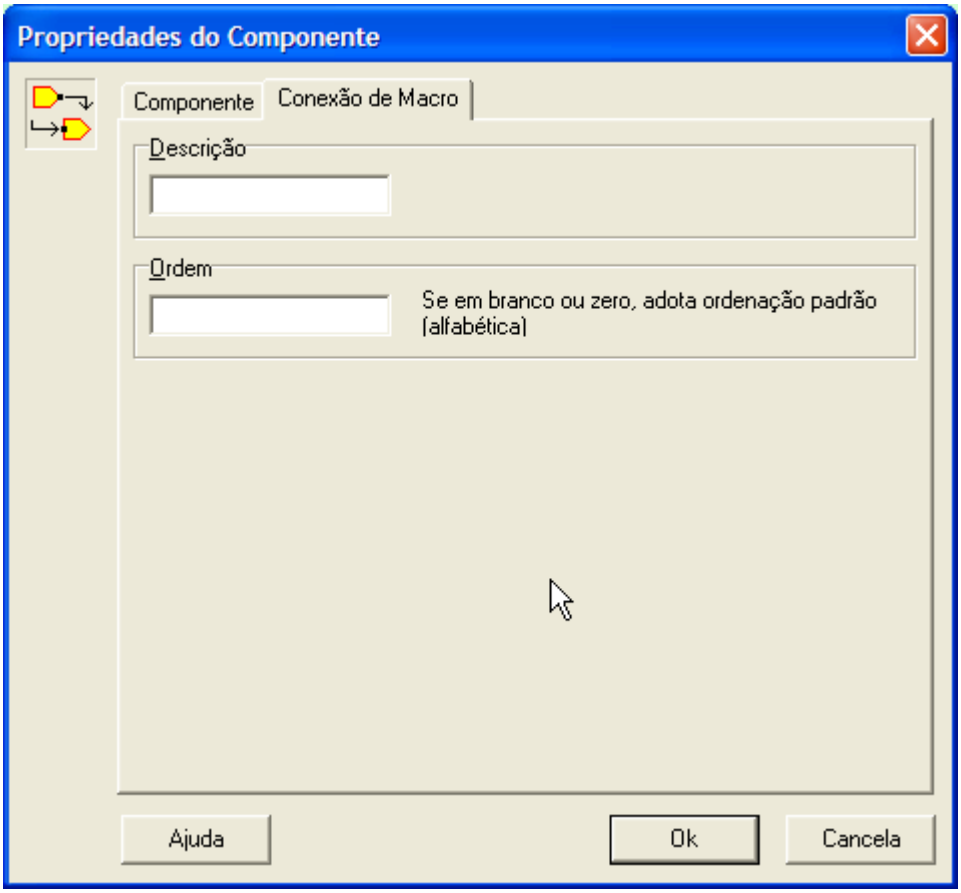


Com isso o programa foi copiado para a página de Macro. Mas todas as variáveis utilizadas em uma Macro não são acessíveis externamente. Portanto, uma Macro como a representada anteriormente não teria nenhuma serventia, pois não teria nenhum contato com o mundo exterior a ela. Vamos, então, utilizar as entidades **Entrada de Macro** e **Saída de Macro** para estabelecer pontos de entrada e saída para a Macro. Neste caso a única entrada seria para o nodo de ativação da Macro, e a única saída poderia ser um nodo indicando que a conversão foi efetuada.

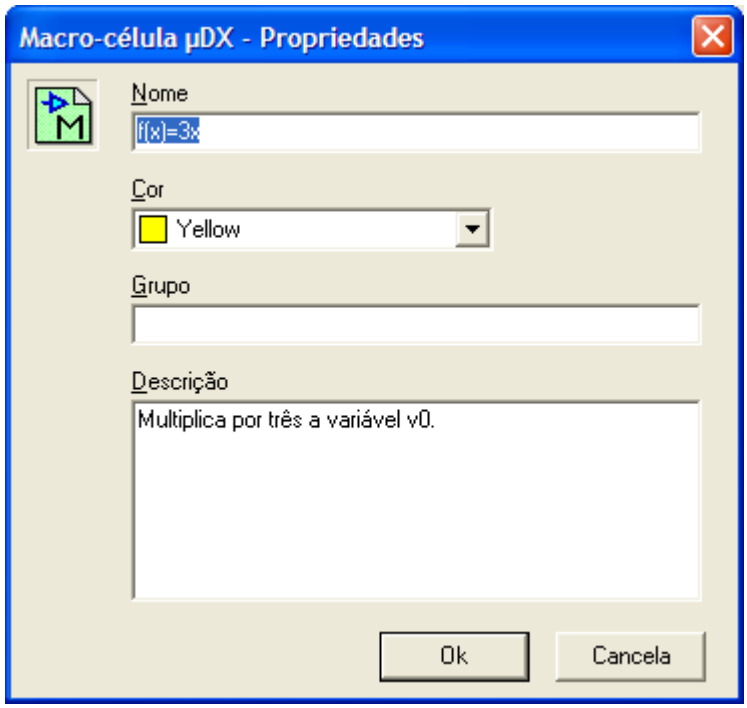
Substituindo o blocode entrada E1 por (Entrada de Macro) e colocando (Saída de Macro) existentes na Barra de Ferramentas do PG (ou vá em [**Editar**] → [**Inserir Entrada de Macro**] e [**Editar**] → [**Inserir Saída de Macro**]), teremos:



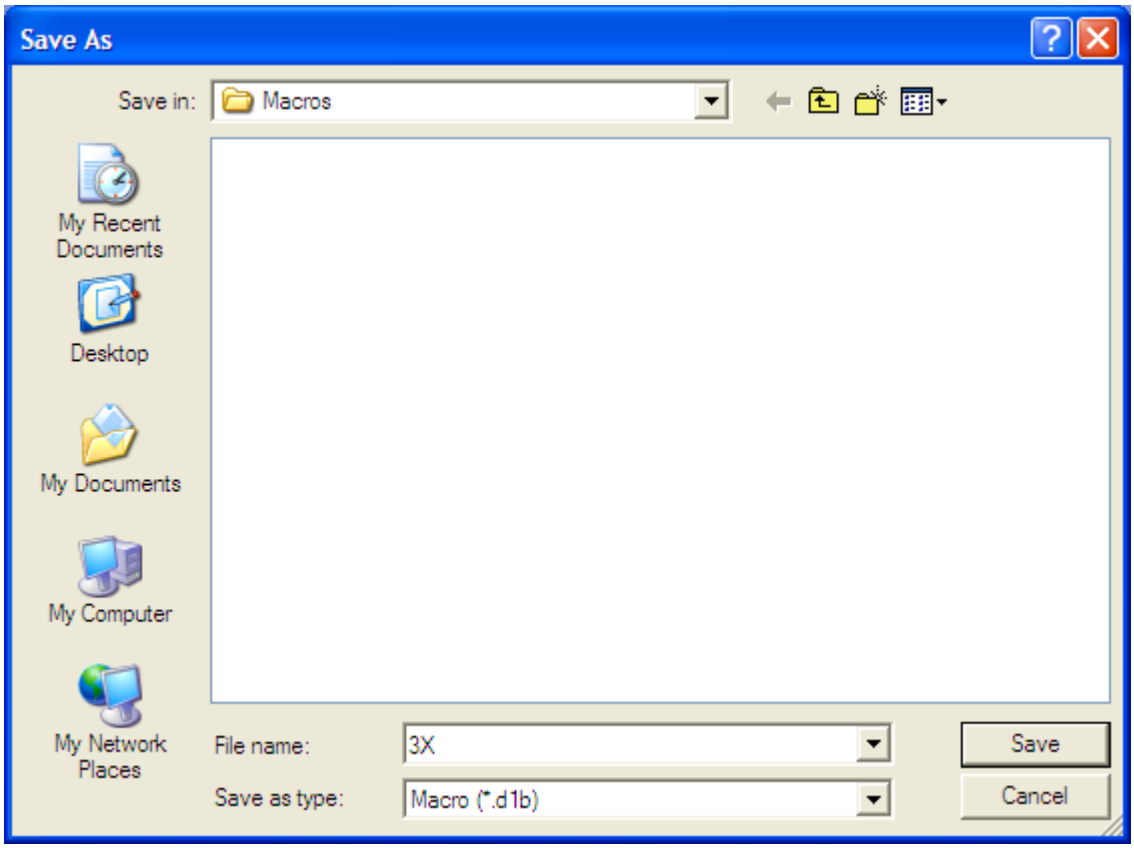
Note que foram editados os blocos de **Entrada de Macro** e **Saída de Macro**, atribuindo nomes a elas. clicando com a tecla direita do mouse sobre as mesmas:



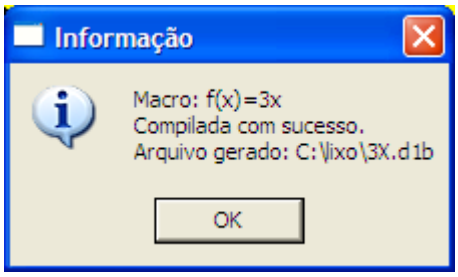
Também editei os blocos para utilizar a variável absoluta v0 como variável de entrada e também como variável de saída do resultado. Como no µDX100 não existe conexão para interligar variáveis preciso usar uma variável absoluta para comunicar uma variável para a macro, já que qualquer nome de variável usado dentro da macro é inacessível ao programa aplicativo, e vice-versa. Está pronta a Macro! Falta apenas selecionar as propriedades da Macro e compilá-la. Para isso vá no menu pop-down [**Macro**] → [**Propriedades da Macro...**]. Digite os seguintes dados (note que foi escolhida a cor amarela para esta macro, mas pode ser escolhida qualquer cor desejada):



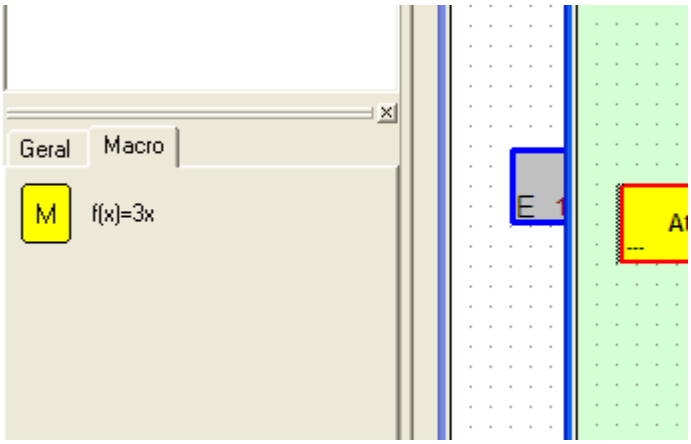
Agora vá em [**Macro**] → [**Compilar e Salvar Macro...**]. O PG irá perguntar o nome da Macro a ser salva. No caso foi chamada de **3X**. Será gerado um arquivo sufixo **D1B** que será o resultado da compilação desta Macro. Estes arquivos devem ser salvos no diretório **Macros** existente no diretório de instalação do PG (normalmente em c:\ Arquivos de Programas \ Dexter \ PG100 \ Macros).



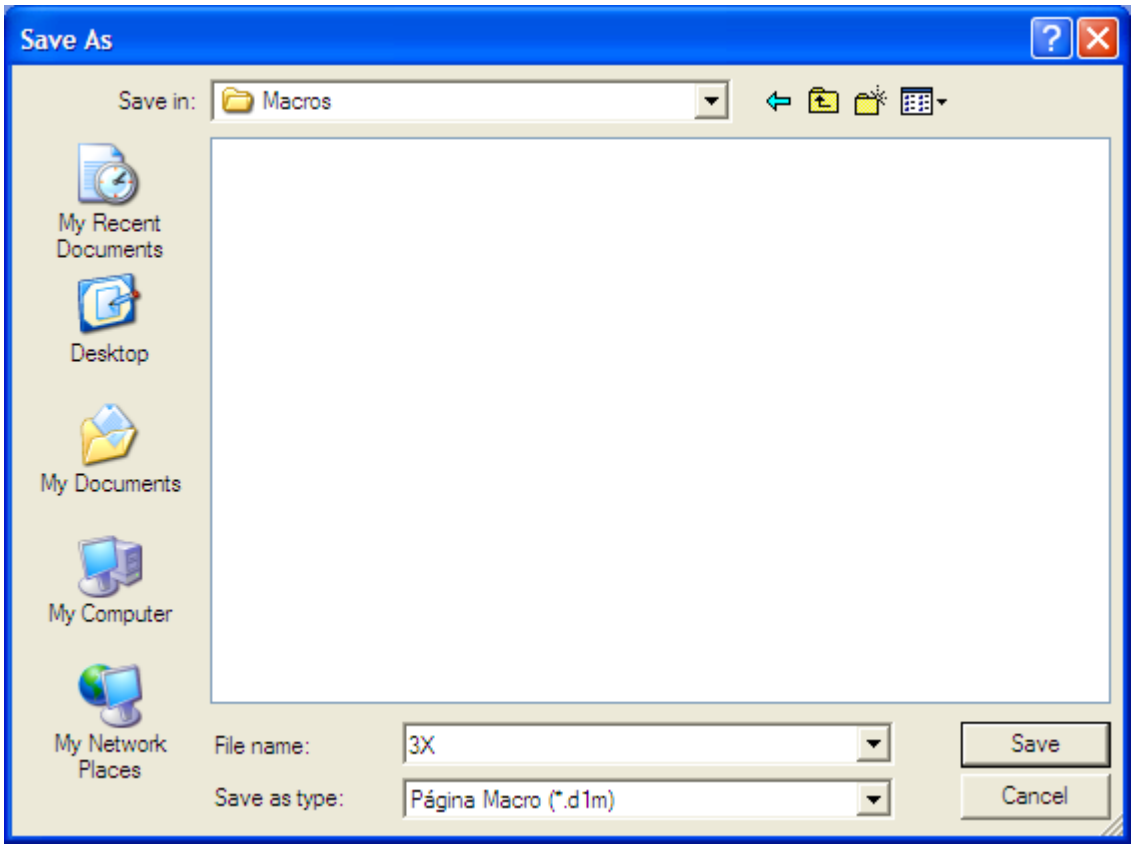
Caso a compilação tenha ocorrido sem falhas deve surgir a seguinte janela de informação:



Note que é gerada uma aba adicional na **Biblioteca de Componentes**, com designação **Macro**, onde já deve constar a macro gerada:



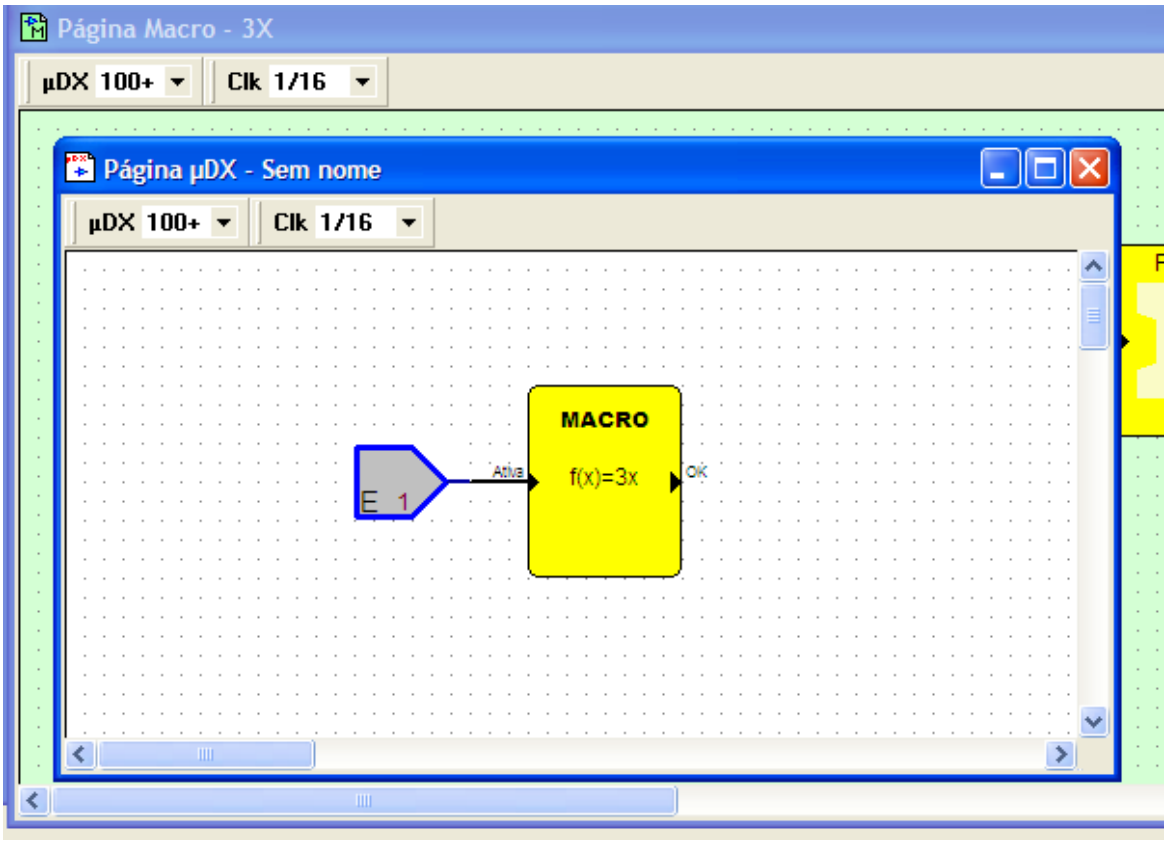
O arquivo **3X.d1b** gerado corresponde a macro compilada. Entretanto, o arquivo fonte desta macro ainda não foi salvo. Para isso clique em na Barra de Ferramentas do PG, ou selecione [**Arquivo**] → [**Salvar**] nos menus.



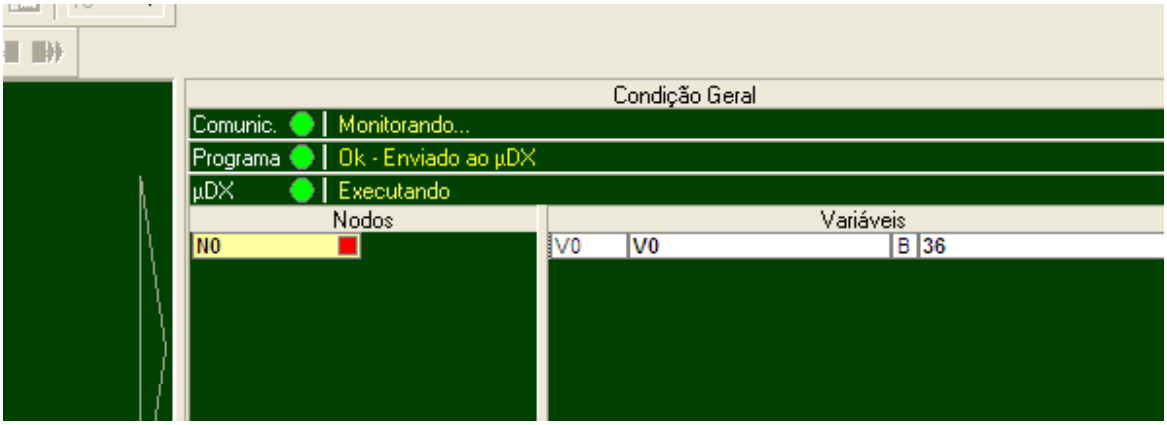
Vamos chamar do mesmo nome (3X). Mas agora será gerado um arquivo sufixo **D1M**, correspondendo ao arquivo fonte da Macro. Com isso, no futuro, será possível implementar modificações nesta Macro. Normalmente se salva fontes de macros no mesmo diretório **c:\ Arquivos de Programas \ dexter \ PG100 \Macros**.

Bem, agora podemos utilizar a Macro criada. Para isso vamos abrir uma nova página µDX clicando em (Nova Página) na barra de ferramentas do PG, ou em [**Arquivo**] → [**Nova Página**]. A seguir, utilize a Macro como se fosse um bloco normal do µDX100 e monte o programa como

a seguir:



Salve a nova página µDX e compile este novo programa, agora usando Macro (use a tecla da barra de ferramentas). No Compilador PG transmita o programa para o µDX100 e teste seu resultado. Abaixo está o resultado para v0=12.

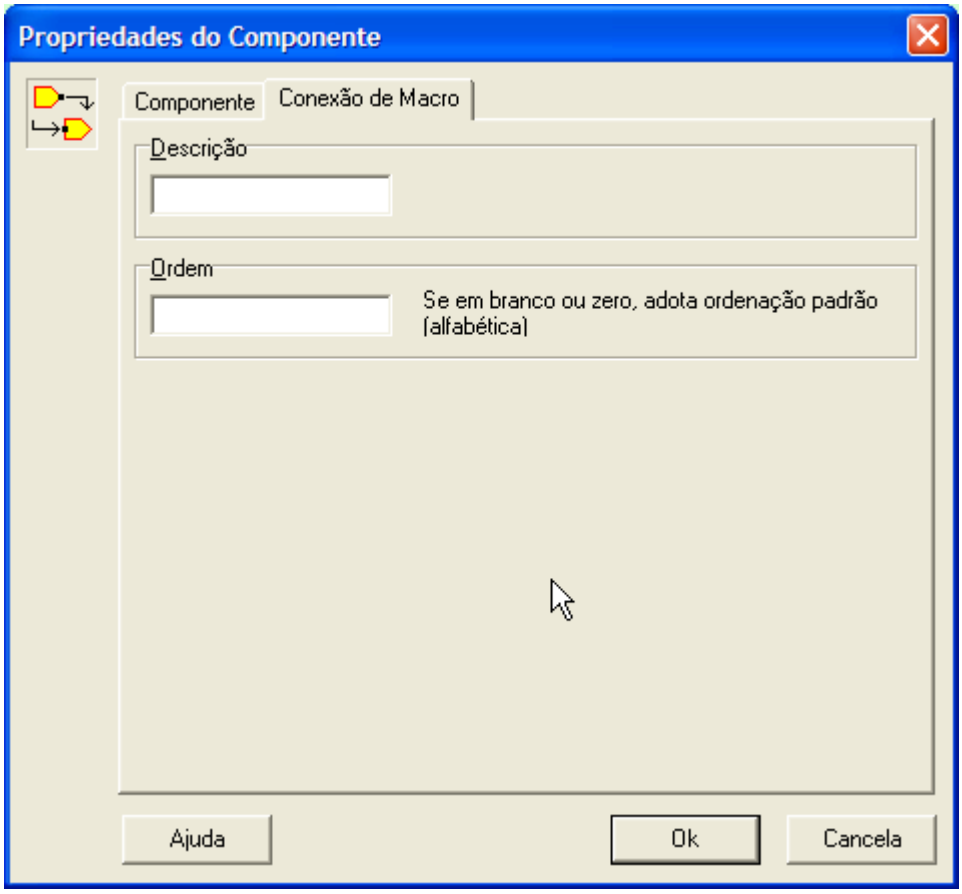


Veja que é perfeitamente possível utilizar a Macro criada na elaboração de outras macros, aumentando gradativamente a complexidade do programa.

Atenção: Apesar da Macro ser representada apenas como um bloco, ela ocupa o número de blocos, variáveis e nodos utilizados para sua elaboração. Assim, o uso massivo de Macros complexas em um programa aplicativo pode, rapidamente, esgotar estes recursos.

Entrada e Saída de Macro

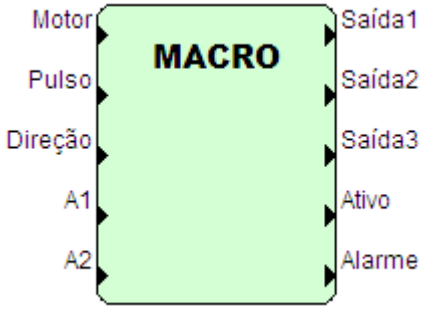
Entrada de Macro e **Saída de Macro** são objetos capazes de inserir pontos de conexão da Macro com o exterior. Estas entradas ou saídas, no caso do µDX100, só podem ser nodos (variáveis binárias). As opções para inserir estes objetos somente ficam ativas quando está selecionada uma página de Macro, já que não é possível usar **Entrada de Macro** ou **Saída de Macro** em programas aplicativos do µDX100. Ao editar uma **Entrada de Macro** ou uma **Saída de Macro** surge a janela abaixo, que permite especificar um nome para a conexão da Macro.



A **Ordem** em uma Entrada de Macro (ou Saída de Macro) especifica qual a ordem em que as conexões de entrada (ou saída) aparecerão no bloco de Macro. As entradas sempre são colocadas à esquerda do bloco de Macro, e as saídas à direita. Com isso, entradas e saídas são ordenadas independentemente. Caso o campo Ordem seja mantido em branco a ordenação será em ordem alfabética crescente. Caso seja colocado números no campo Ordem, os itens serão ordenados em ordem crescente destes números. É aceitável qualquer número inteiro (portanto, é possível usar-se números negativos). Se duas os mais entradas de Macro possuírem o mesmo número de ordem, elas serão ordenadas alfabeticamente. Por exemplo, digamos que foi gerada uma Macro com as seguintes conexões de entrada e saída:

Entradas de Macro		Saídas de Macro	
Descrição	**Ordem**	**Descrição**	**Ordem**
A1		Saída1	-3
A2		Saída2	-3
Motor	-1	Saída3	-3
Pulso	3	Alarme	10
Direção	4	Ativo	2

Neste caso o PG irá gerar um bloco de Macro como mostrado a seguir:



Convertendo programas em Ladder

"LADDER" é uma linguagem de programação para Controladores Programáveis comumente empregada por indústrias e empresas de automação.

Inteiramente baseada nos métodos de projetos elétricos para controles por relés e circuitos discretos, a "LADDER" carregou junto com esta compatibilidade de conceitos uma complexidade de estrutura, precisando de especialistas para seu bom aproveitamento.

A linguagem PDE (Programação por Diagrama Esquemático), que lembra um circuito elétrico convencional - e, portanto, mais fácil de ser interpretada - pode ser empregada para reproduzir um programa feito em "LADDER".

A programação em "LADDER" atende a algumas convenções:

- Existe uma barra de energia (+V) à esquerda e uma barra de 0V à direita do circuito desenhado.
- As conexões do circuito sempre obedecem a este sentido de fluxo de energia: para ligar um relé ele precisa ter uma extremidade conectada (diretamente ou através de outros circuitos) à barra de energia e a outra extremidade conectada na barra de 0V.
- Um relé pode ter diversos contatos auxiliares.
- O retorno de algum sinal, como a saída de um temporizador para sua própria entrada (formando um oscilador), somente pode ser feita através de um relé (acionado pelo temporizador) e um contato auxiliar deste relé conectado na entrada do temporizador.

Em PDE estas convenções podem ser esquecidas ou replicadas.

As barras de energia e de 0V não são necessárias, uma vez que a energia (simbolizada pela figura de pilhas ou resultante de alguma entrada ativada) pode ser aplicada em qualquer local do esquema desenhado. O nível de 0V (zero volts) está implícito nas conexões em aberto ou no símbolo aparente no desenho dos relés de saída.

O sentido do fluxo de corrente, da esquerda para a direita, é importante apenas nas próprias instruções (como as chaves e outros blocos). As conexões feitas pelos fios desenhados podem dar voltas à vontade em qualquer região do desenho, como ocorre em um diagrama esquemático convencional.

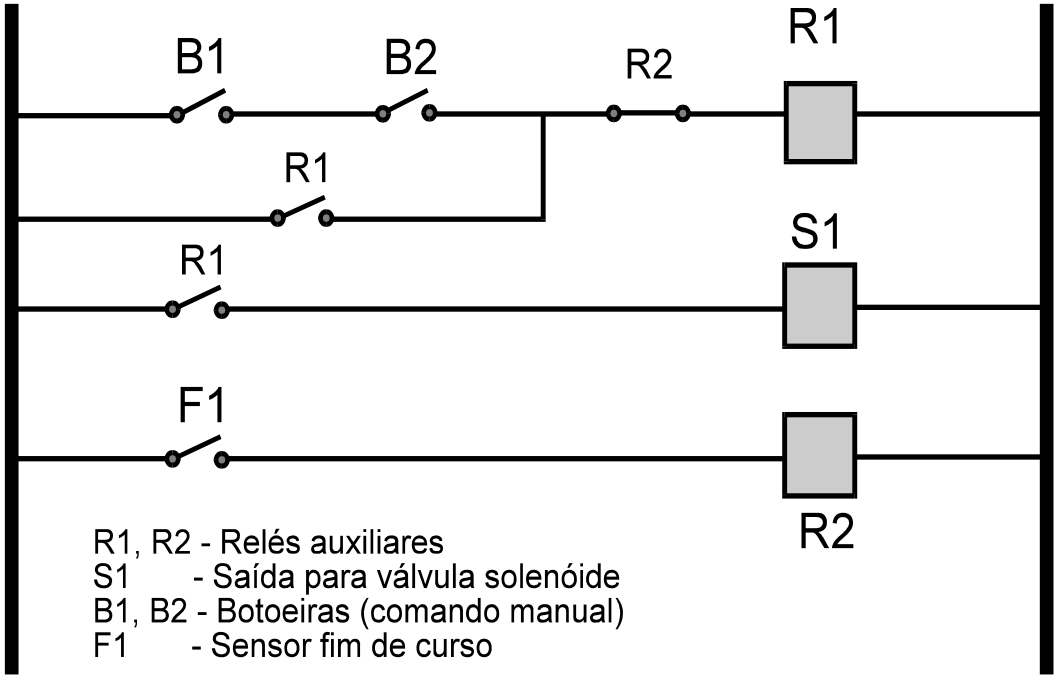
Os relés e contatos auxiliares empregados na linguagem "LADDER" podem ser representados pelas chaves (NA e NF) do µDX. O controle de uma chave funciona como a bobina do relé auxiliar e os contatos da chave como os contatos do relé.

Assim como em "LADDER" um relé pode ter mais de um contato auxiliar, também em PDE pode-se implementar esta idéia utilizando-se mais de uma chave e unindo-se os controles para serem atuadas todas de uma vez só.

É importante entender que no uso das chaves do µDX100 o controle é uma conexão (elétrica) atuada por alguma das entradas do µDX ou por uma saída de algum bloco de função. Assim, uma conexão em PDE pode representar um relé em "LADDER".

O desenho a seguir mostra um pequeno programa feito em "LADDER" que serve para controlar a lógica de operação de uma prensa (hipotética). As botoeiras B1 e B2 servem para iniciar a operação e precisam ser acionadas juntas (na prática elas devem ser montadas distantes de forma que o operador precise das duas mãos para pressionar as botoeiras). Uma vez iniciada a operação o relé auxiliar R1 é ligado (já que o contato auxiliar do relé R2 é normalmente fechado). O primeiro contato auxiliar de R1 irá manter o próprio R1 ligado mesmo que as botoeiras sejam liberadas (travamento). O segundo contato auxiliar de R1 energiza o relé S1 que, na prática, deve atuar uma válvula solenóide. Esta válvula comanda o movimento da prensa.

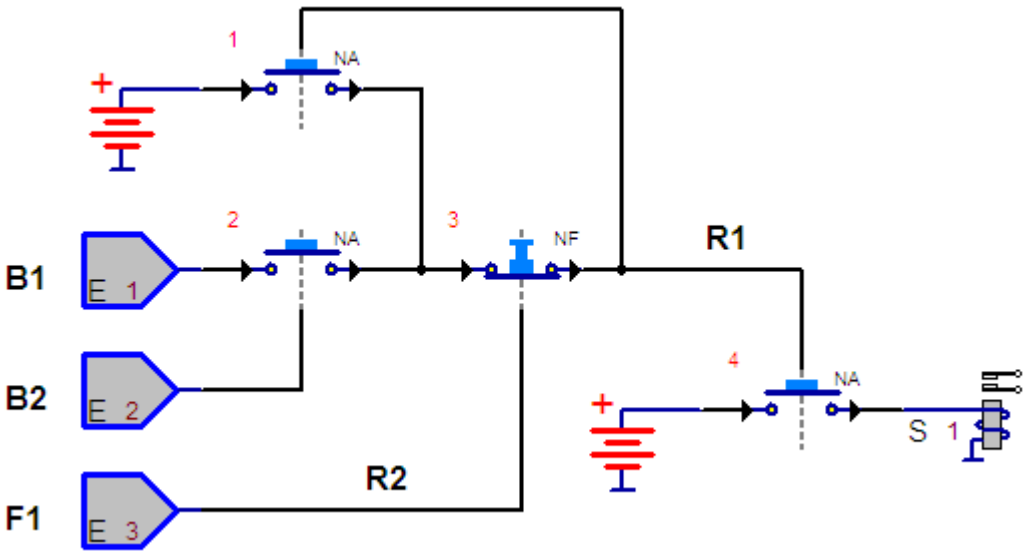
No fim do movimento da prensa um sensor tipo fim-de-curso é atuado ligando o relé auxiliar R2. Este relé, quando ligado, abre seu contato auxiliar desligando o relé R1, que desligará a saída S1, com a lógica toda voltando ao estado inicial.



O próximo desenho mostra o equivalente em PDE ao programa de controle da prensa. As conexões que funcionam como os relés auxiliares R1 e R2 estão apontadas (note que este apontamento foi feito apenas para fins didáticos e não é necessário para o funcionamento do programa).

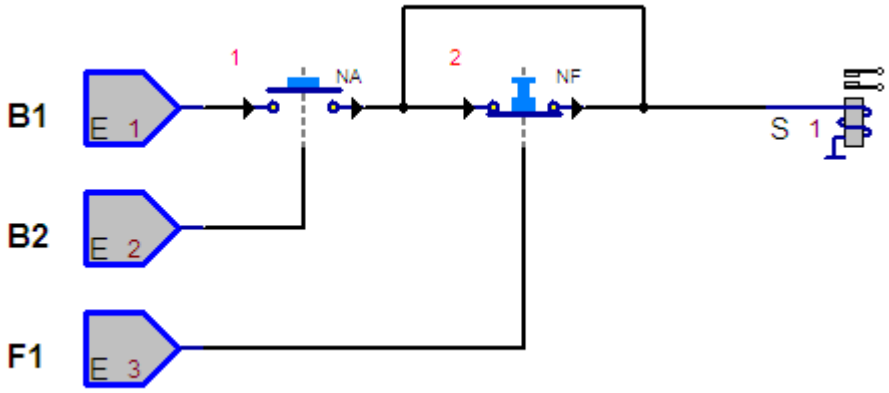
O símbolo de bateria foi utilizado nos pontos onde é preciso haver energia para a atuação do circuito. As entradas E1 e E2 (para as botoeiras) efetuam a lógica "AND" através da chave NA. A outra chave NA acima é utilizada para funcionar como o contato de travamento (retenção), suprindo de energia a conexão antes da chave NF caso alguma das entradas fique inativa (e após "R1" ter sido ligado).

A entrada do sensor fim-de-curso conecta-se diretamente no controle da chave NF, uma vez que estando E3 ativa o relé "R2" deve desligar o relé "R1".



Exemplo de Programa Aplicativo: **Ladder.d1g**

Pode-se simplificar ainda mais o esquema acima, utilizando uma particularidade da linguagem PDE. A retenção pode ser feita por realimentação direta do sinal de saída, sem a necessidade da chave NA superior. Neste caso, o esquema seria o representado a seguir. Note que utilizei o esquema de retenção via chave NF (normal fechada). Ao energizar a entrada da chave NF ela se auto-sustenta pela realimentação da saída da chave NF. Ao energizar a linha de controle da chave NF abre-se a porta NF, interrompendo a auto-sustentação e desligando a linha de saída.



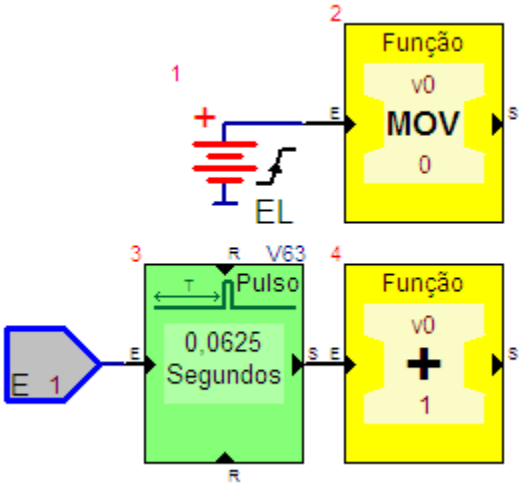
Exemplo de Programa Aplicativo: **Ladder2.d1g**

Portanto, o esquema final em PDE fica como mostrado acima. Pode-se perceber uma simplificação em relação à linguagem "LADDER", pois não há necessidade dos relés auxiliares.

Após compilar o programa e transmití-lo ao µDX100, mande-o executar este programa exemplo. Inclua os nodos n0, n1 e n2 na lista de nodos monitorados. Apontando com o mouse para o nodo n0 (correspondente a entrada E1) e pressionando a tecla esquerda do mouse é possível acionar esta entrada. Fazendo o mesmo com E2 (nodo n1) a chave NA se fecha, energizando o relé de saída S1. Aponte para E1 e E2 e pressione novamente a tecla esquerda do mouse, desativando estas entradas. Apesar de desativadas, a saída continuará energizada devido a auto-sustentação. Entretanto, ao ativar a entrada E3 (nodo n2) a chave NF se abre, interrompendo a auto-sustentação e desligando a saída S1.

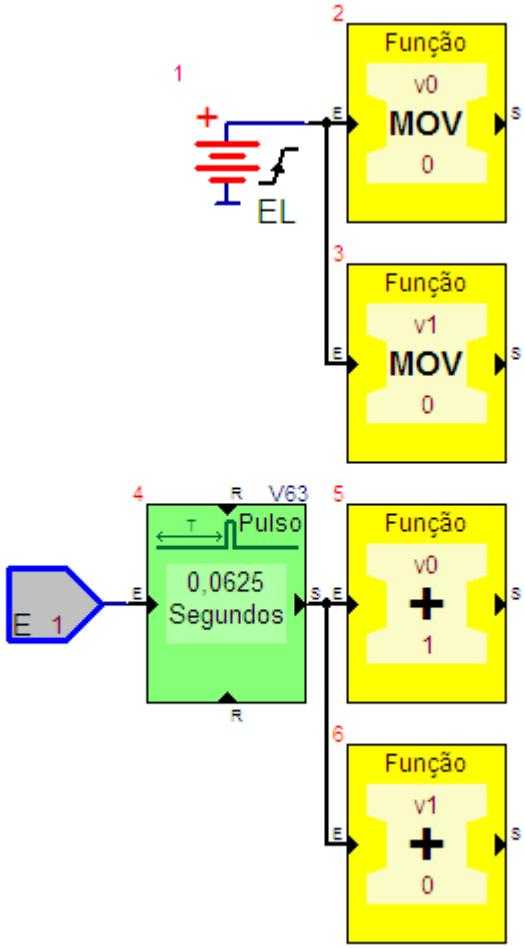
Operações Aritméticas com mais de 8 bits

O controlador µDX100 utiliza variáveis com 8 bits, sem sinal. Assim, o máximo valor que estas variáveis podem asssumir é 255 (2^8). Entretanto, muitas vezes pode ser necessário manipular valores numéricos maiores. Para isso, a versão 6.0 do µDX ou superior prevê a propagação de estouro (carry ou borrow) no caso de operação de adição ou subtração. Neste caso, utiliza-se duas funções anteriormente inúteis, que eram somar ou subtrair 0 (zero) de uma variável. Quando é efetuada uma operação de soma ou subtração em que ocorre estouro, esta informação é armazenada pelo µDX e, no primeiro bloco com adição com zero ou subtração com zero este estouro é adicionado ou subtraído da variável. Por exemplo, se tivermos um bloco que incrementa v0 a cada energização da entrada E1 (para contar número de peças que passam por um sensor ligado a E1), o programa poderia ser como abaixo (note o bloco PULSO para garantir apenas um incremento para cada energização de E1, e o bloco NODO EL para inicializar a variável v0 com valor zero):



Exemplo de Programa Aplicativo: **Aritmetica_8bits.d1g**

Este programa incrementa v0 a cada energização da entrada E1. Mas, após 256 energizações, a variável v0 volta ao valor zero (pois 255+1 = 256 -> estouro de variável). Então, pode-se usar uma segunda variável para continuar a contagem (usando a propagação de carry via bloco com adição com zero):



Exemplo de Programa Aplicativo: **Aritmetica_8bits2.d1g**

Note que, quando houver estouro em v0, o carry será considerado no bloco subseqüente (v1+0), incrementando v1. Assim, o valor de contagem será obtido pela fórmula:

Contagem = 256 * v1 + v0

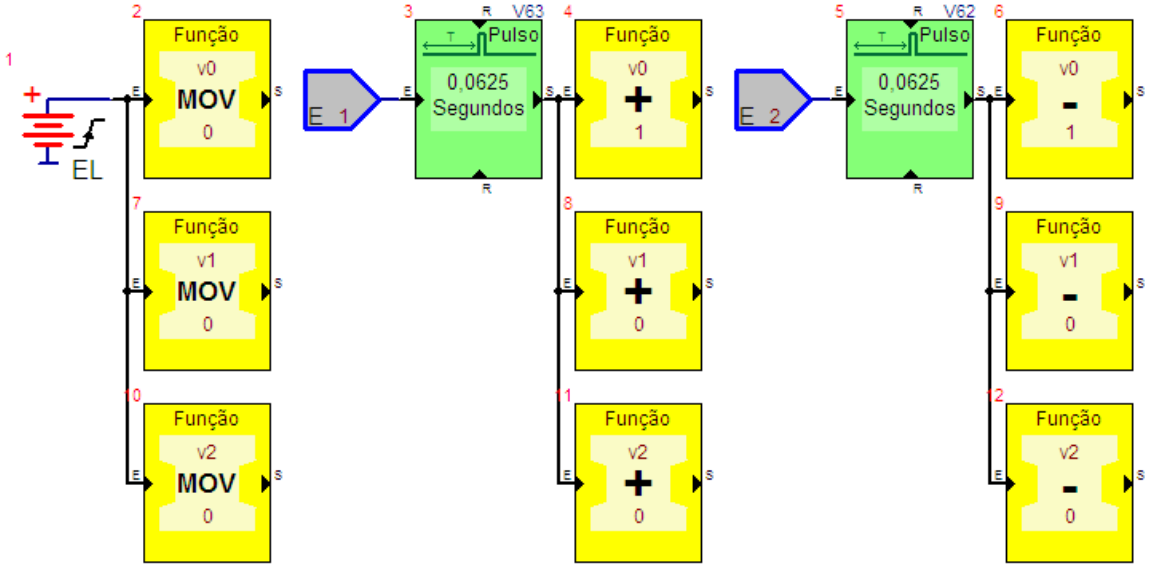
O valor máximo atingível é de 65535 contagens. Se ainda assim não for o suficiente basta adicionar mais variáveis. Neste caso, a contagem será:

Contagem = 256 * 256 * v2 + 256 * v1 + v0

O limite, neste caso, é de 16777215 contagens!

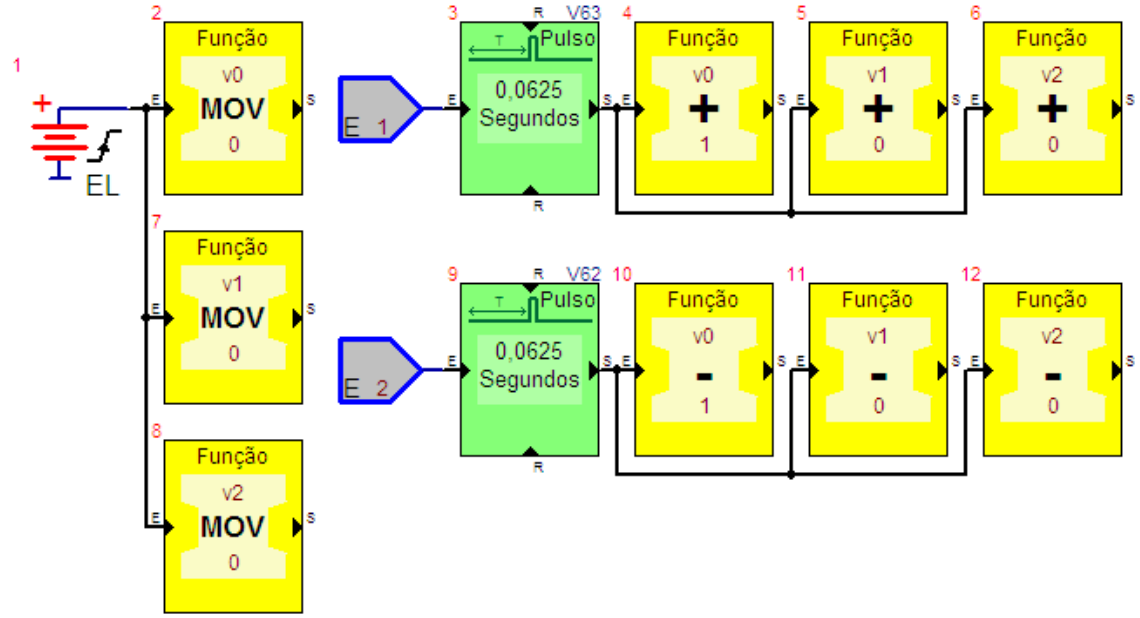
Atenção: o bloco de função soma com zero sempre irá considerar o carry do bloco imediatamente anterior. Assim, os blocos responsáveis pelo cálculo com mais de 8 bits devem ser posicionados em seqüência no programa aplicativo. No exemplo anterior estes blocos seriam os blocos 5 e 6 (em seqüência, portanto, correto).

O mesmo raciocínio pode ser aplicado para a operação de subtração. O programa abaixo exercita esta facilidade. A entrada E1 incrementa a contagem, enquanto a entrada E2 decrementa a mesma contagem.



Exemplo de Programa Aplicativo (incorreto): **Aritmetica_8bits3.d1g**

Mas o programa acima não irá funcionar corretamente! Isso porque o estouro de carry exige que os blocos de função sejam subseqüentes, e se nota claramente que para adição os blocos ficaram com numeração 4, 8 e 11. Já para subtração os blocos são 6, 9 e 12. Para corrigir isso basta posicionar os blocos convenientemente, tendo em mente que o PG sempre numera os blocos da esquerda para a direita, e de cima para baixo (como se lê um livro):



Exemplo de Programa Aplicativo (correto): **Aritmetica_8bits4.d1g**

Atenção: A partir da versão 9.6 do controlador µDX101 já é permitida operação direta em word (16 bits sem sinal), de forma que o recurso descrito acima é desnecessário.

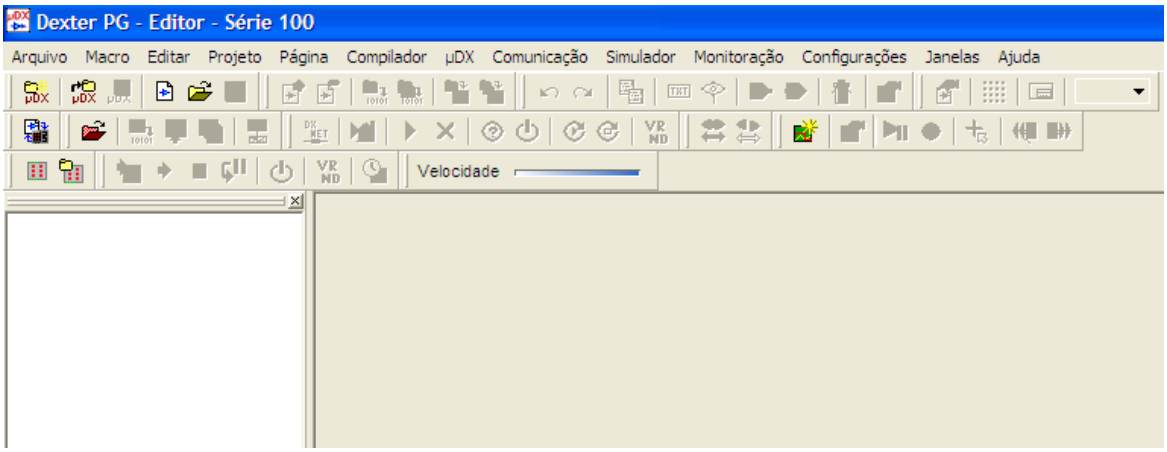
Parte VIII

Simulador

A partir da versão 2.0.0.0 do software PG (Programador Gráfico) para µDX100 está disponível um simulador, de forma que é possível testar programas aplicativos sem a presença do controlador programável. Isso permite depurar problemas de lógica com mais facilidade, pois o Simulador possui facilidades, como a execução passo a passo.

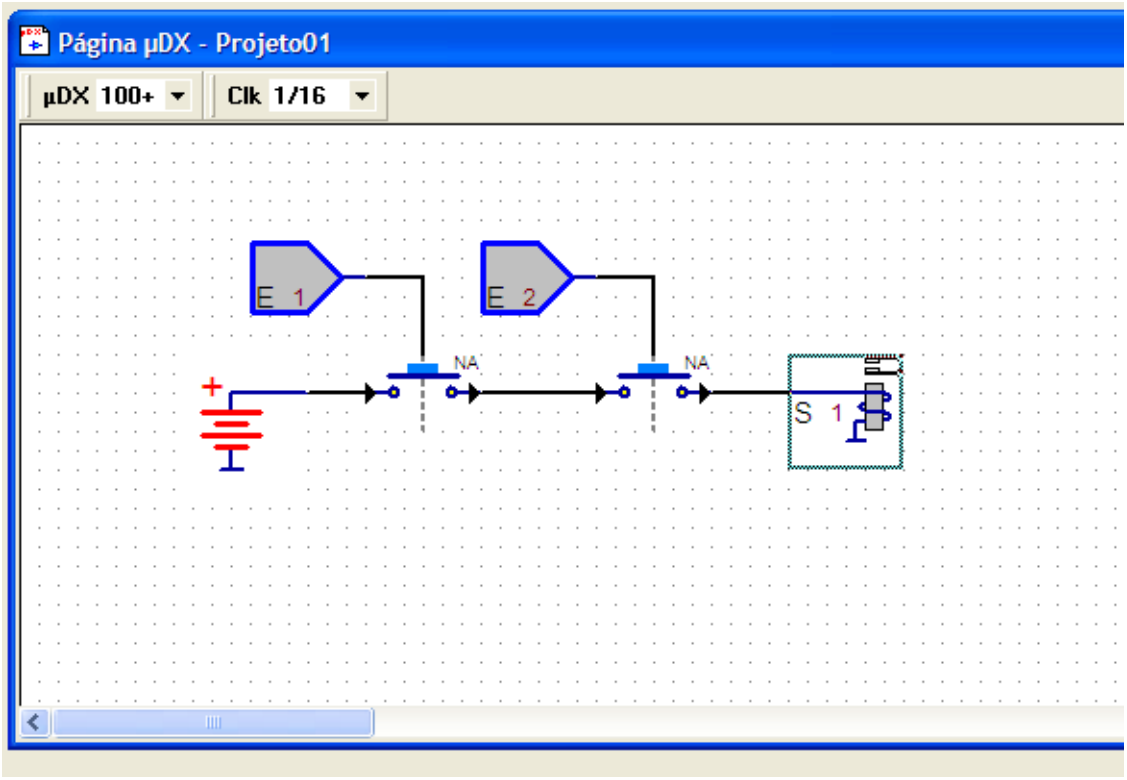
O Simulador possui uma interface bastante similar à do Compilador, de forma a facilitar seu uso. Para diferenciá-los rapidamente, o Simulador usa fundo azul em vez do fundo verde do Compilador.

Todas as principais funções do Simulador estão disponíveis na terceira barra de ferramentas, como mostrado na figura abaixo. Também o menu pop-down "Simulador" permite acessar estas funções, e algumas adicionais.



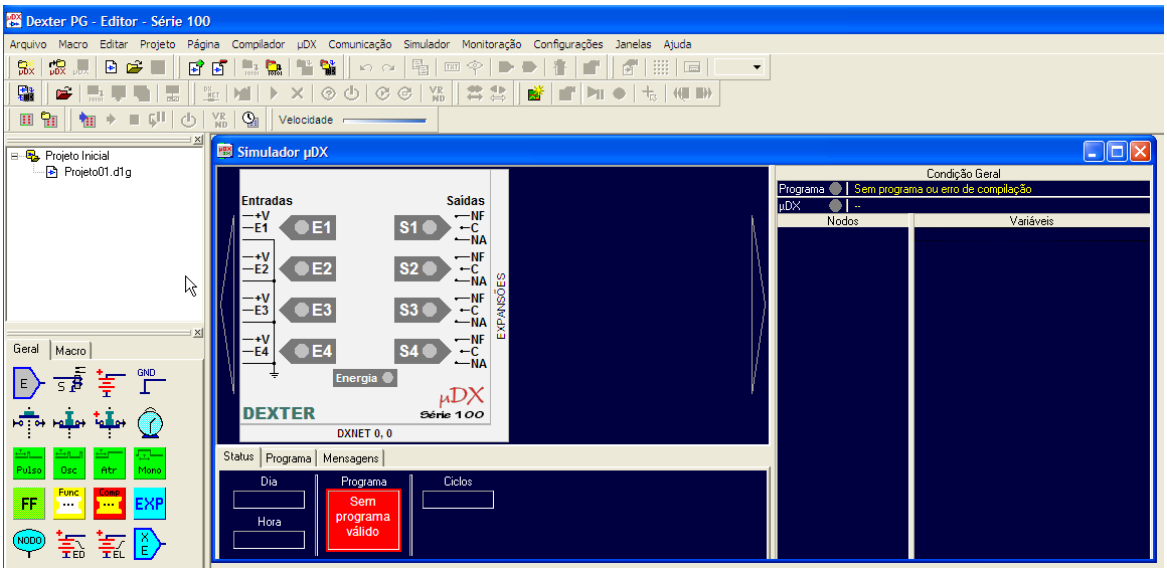
Para simular um programa primeiramente é preciso carregá-lo no PG. Para isso pode ser usada a tecla  (Abrir página...) para carregar um programa aplicativo. Caso já exista um programa aplicativo carregado no PG e tecla  (Simular Programa) para compilar o programa aplicativo e transferí-lo para o Simulador.

Digamos que seja usado o projeto gerado na seção **Elaborando Programas** para testarmos o Simulador. Para isso carregue o projeto **Teste1.d1p**, conforme mostrado abaixo:

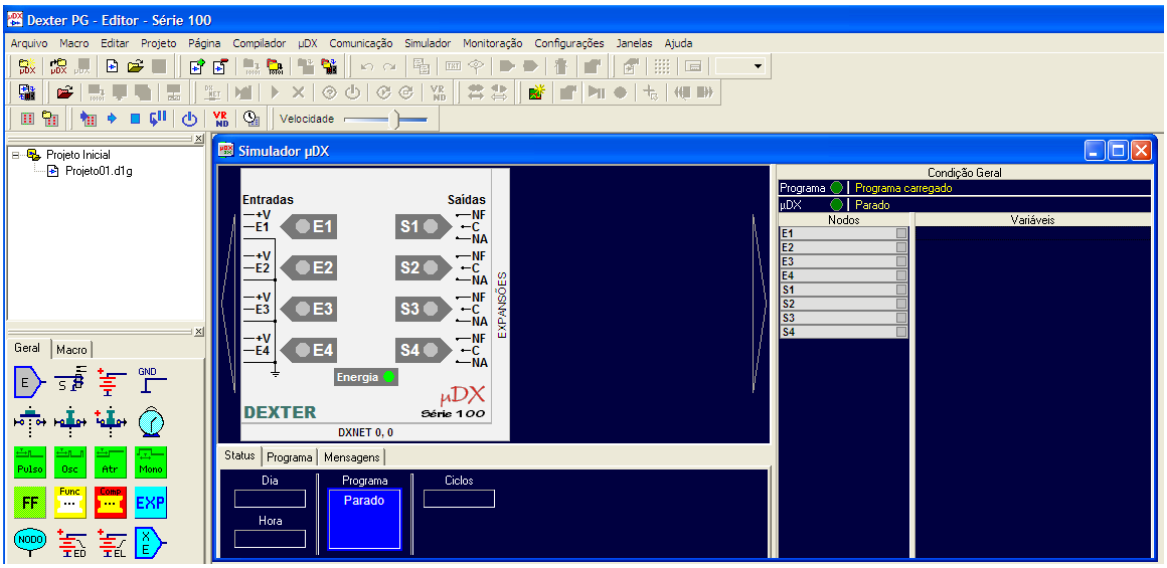


Exemplo de Projeto no PG: **Teste.d1p** (projeto); **Teste.u1p** (compilado).

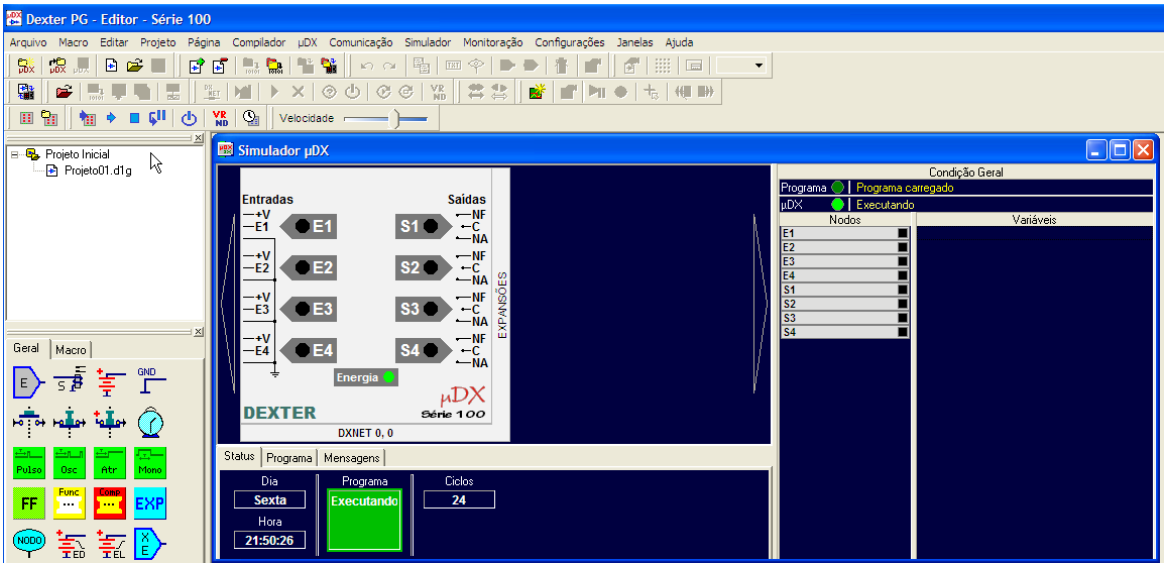
Agora pressione a tecla (Simulador/fontes) existente na barra de ferramentas do Simulador. Note que o PG inclui três linhas na barra de ferramentas: a primeira se refere ao Editor, a segunda ao Compilador, e a última é referente ao Simulador. Com isso será aberto o Simulador, conforme a figura a seguir:



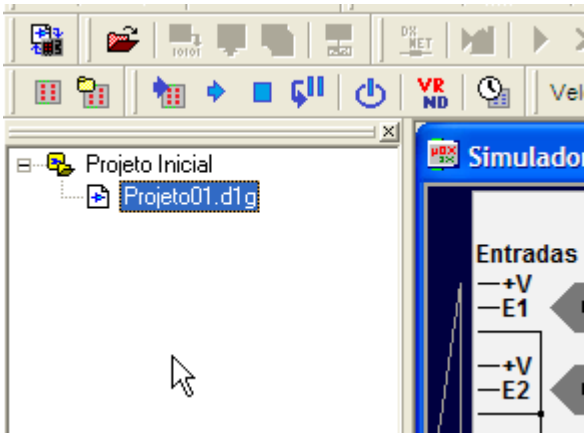
Note que na área em vermelho é indicado não existir ainda programa válido no Simulador. Para transferir o projeto **Teste1.d1p** para o Simulador pressione a tecla (Simular Programa). Com isso obtemos a tela a seguir:



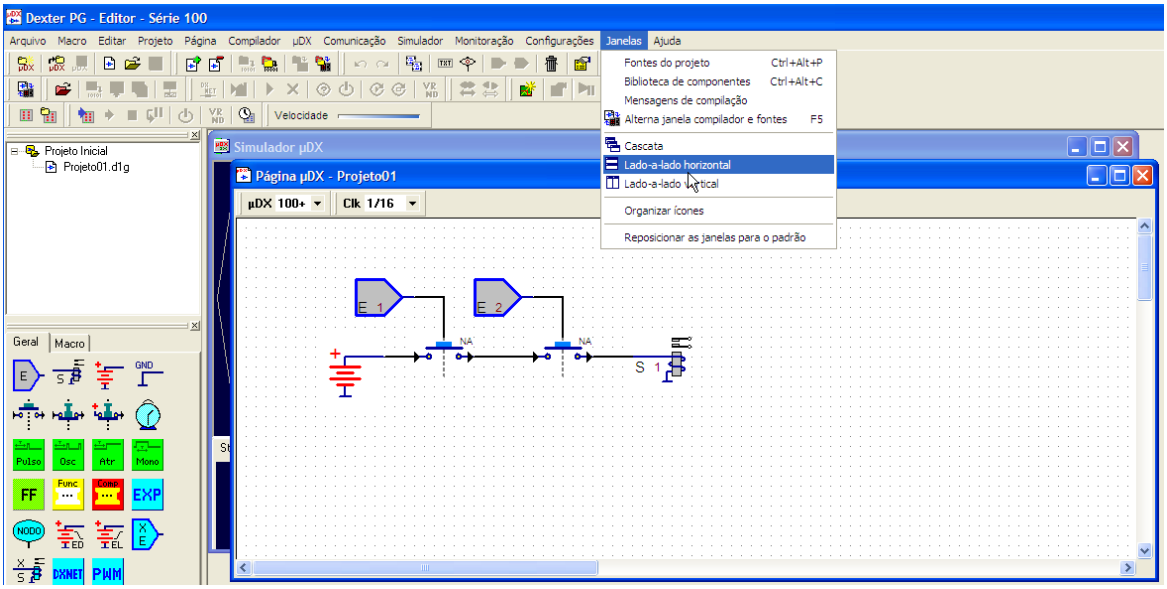
Agora já estamos no Simulador com o programa aplicativo carregado e pronto para simulá-lo. A tecla (Rodar) permite iniciar a simulação. A tecla (Parar) interrompe a simulação, e a tecla (Passo-a-passo) possibilita executar a simulação passo a passo. Pressione a tecla (Rodar). O contador de ciclos será incrementado a cada ciclo de simulação do programa aplicativo:



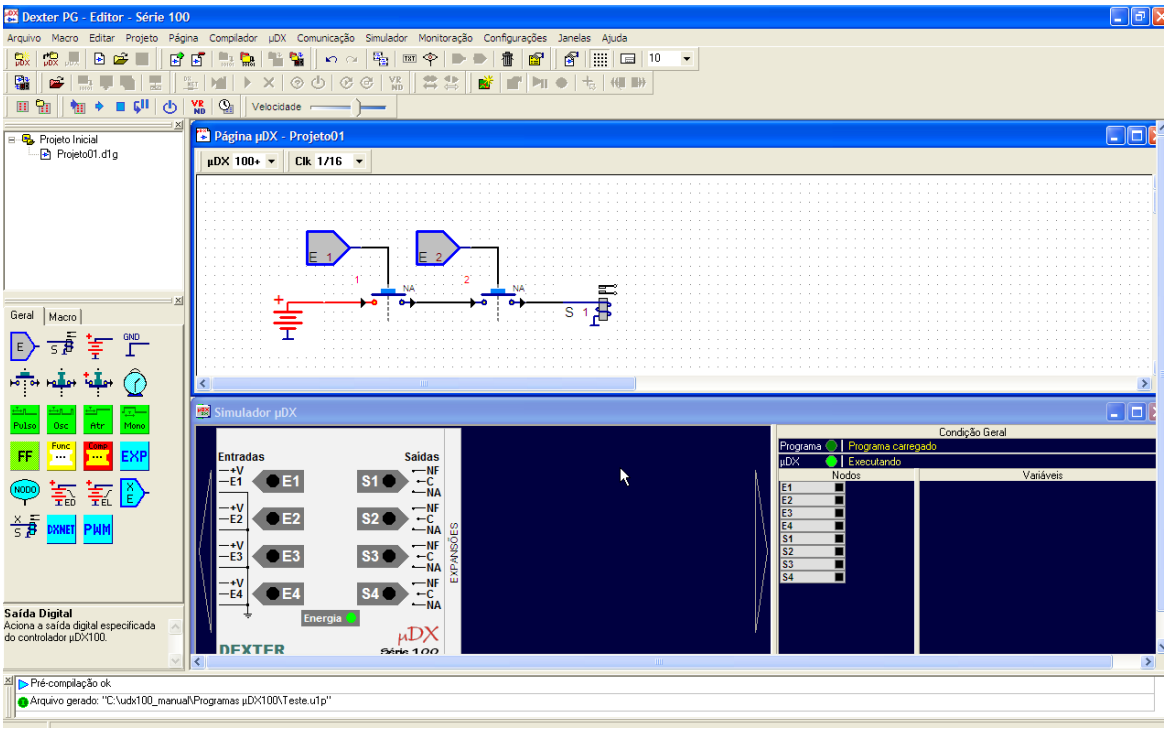
Pode-se visualizar simultaneamente páginas de edição do programa aplicativo e a tela do Simulador. Para isso abra a página Projeto01.d1g pressionando duas vezes a tecla esquerda do mouse sobre o nome da página no canto superior esquerdo da tela do PG:



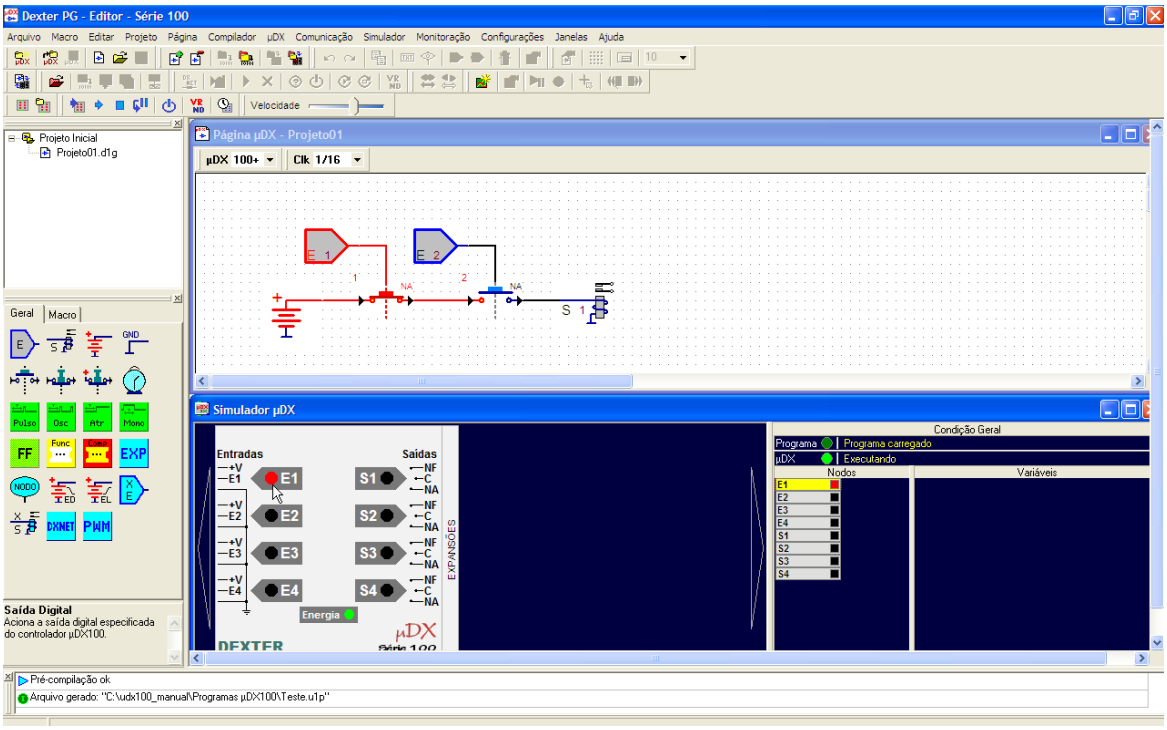
Irá surgir a página sobreposta à simulação. É possível ordenar as páginas manualmente ou, por exemplo, escolher a opção **Lado-a-lado horizontal** no menu pop-down **Janelas**:



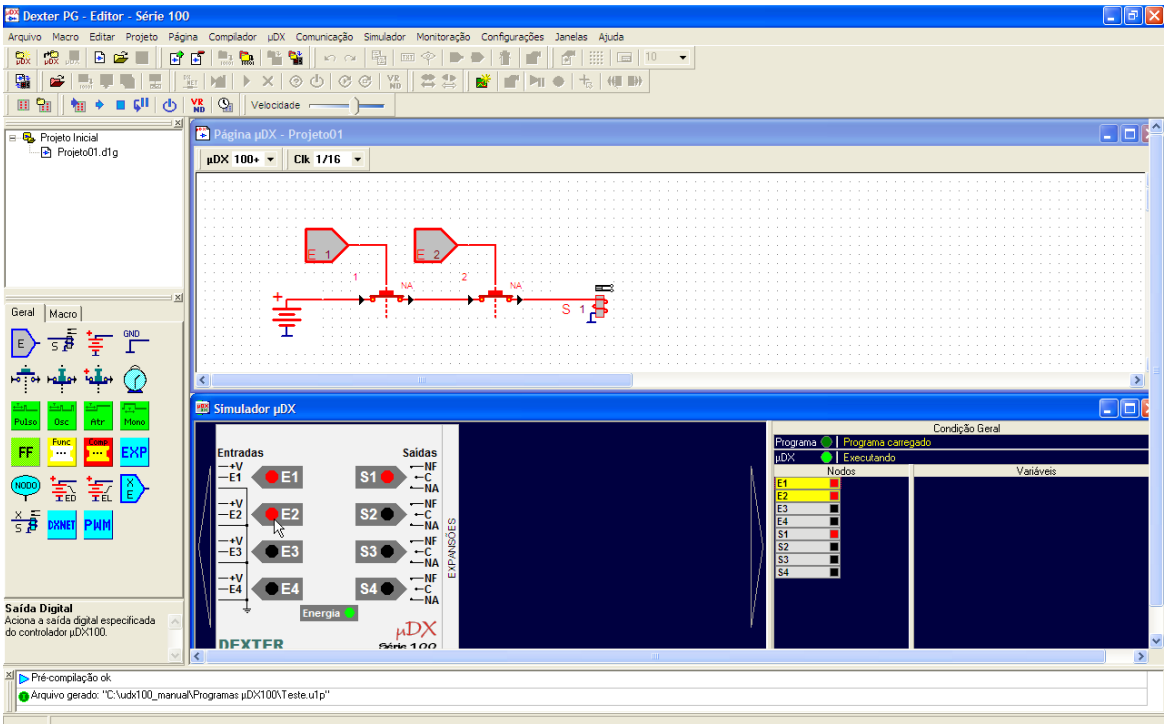
O resultado obtido será o seguinte:

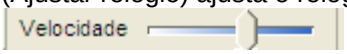


Note que o programa aplicativo mostra as linhas energizadas em vermelho, e os blocos cujas saídas estão ativas também são apresentados em vermelho. Isso facilita bastante a depuração dos programas. Para acionar uma entrada basta clicar duas vezes com a tecla esquerda do mouse sobre a representação do µDX100, ou sobre o nodo que representa a entrada. Por exemplo, ao acionar a entrada E1 a chave NA controlada por esta entrada é fechada, permitindo que a energia chegue à entrada da segunda chave:



Ao clicar sobre a entrada E2 a segunda chave NA é acionada, e com isso a saída S1 é ligada:



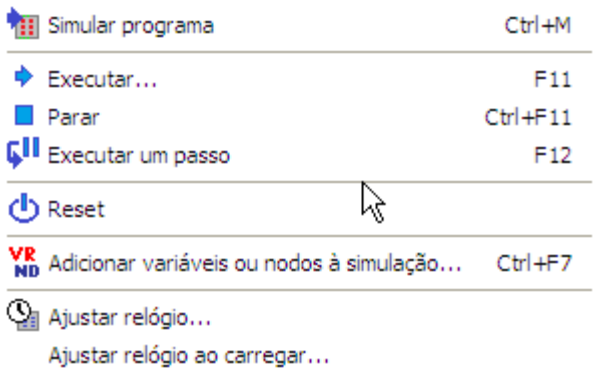
A tecla (Reset) permite reinicializar a simulação. Já a tecla (Adicionar variáveis ou nodos) possibilita inserir nodos ou variáveis à lista de nodos e variáveis simulados. A tecla (Ajustar relógio) ajusta o relógio de tempo real no µDX100 simulado. Por fim, a barra rolante 

permite ajustar a velocidade da simulação.

Nota: Para que os blocos tenham representação diferenciada durante a simulação (chaves abrindo e fechando, blocos em vermelho, etc) é imprescindível que o programa aplicativo tenha sido gerado com biblioteca versão 1.1 ou posterior. Programas gerados com biblioteca versão 1.0 têm blocos sem representação diferenciada e, por isso, não modificam seu aspecto durante a simulação.

Menu Simulador

O menu Simulador permite acionar o simulador, executar e parar o simulador, executar passo a passo, reinicializar a simulação, adicionar variáveis e nodos à simulação, e ajustar o relógio do µDX simulado.



Simular programa (Ctrl+M)

Abre o Simulador e transmite o programa aplicativo para o Simulador. Note que a simulação sempre inicia parada.

Executar (F11)

Inicia a simulação. O contador de ciclos é incrementado a cada ciclo de execução do programa aplicativo.

Parar (Ctrl+F11)

Interrompe a simulação.

Executar um passo (F12)

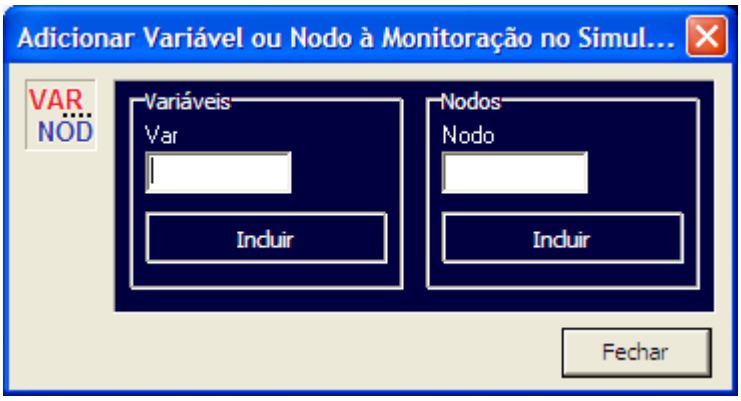
Executa apenas um ciclo de execução do programa aplicativo. Esta é uma ferramenta poderosa para depurar problemas, pois permite examinar passo a passo o comportamento do programa aplicativo

Reset

Reinicializa a simulação. Equivale a efetuar um reset no controlador programável.

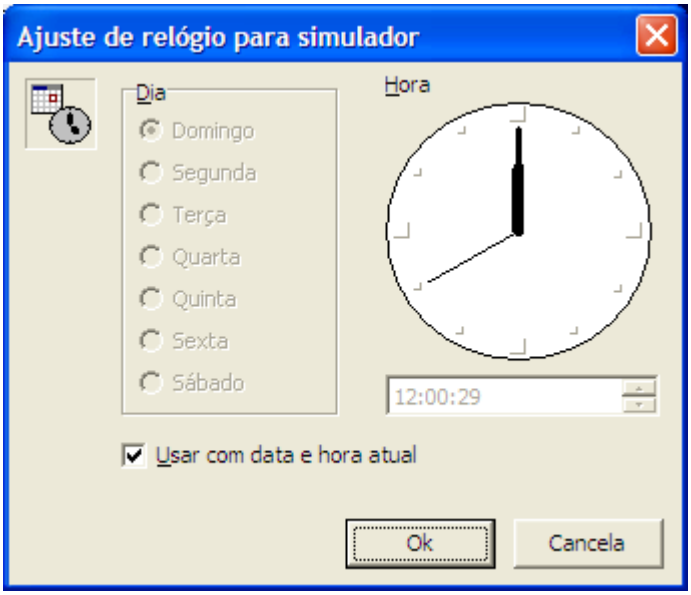
Adicionar variáveis ou nodos à simulação... (Ctrl+F7)

Permite acrescentar variáveis e nodos à simulação. Note que apenas nodos e variáveis devidamente nomeados no programa aplicativo são visualizados na simulação. Com este recurso qualquer outro nodo ou variável pode ser visualizado.



Ajustar relógio...

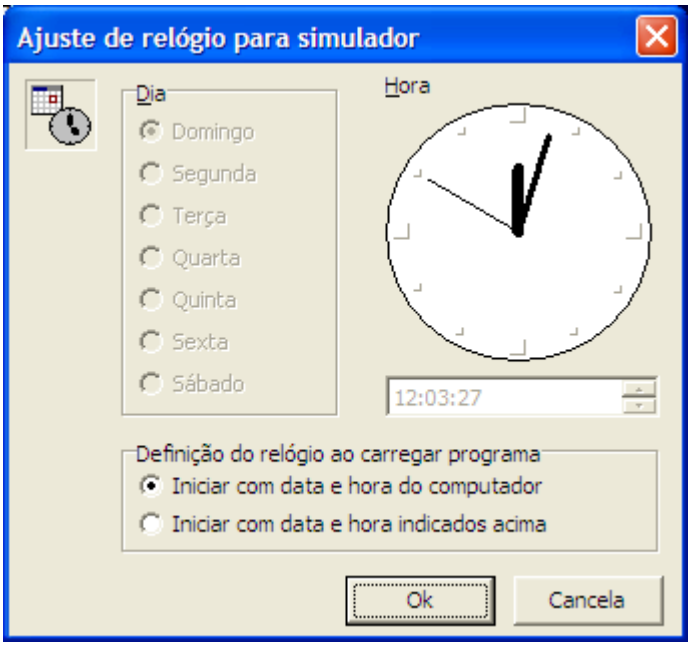
Ajusta o relógio de tempo real da simulação de controlador µDX, permitindo testar programas que sejam dependentes do horário (usem o bloco **Relógio**).



Caso a opção **Usar com data e hora atual** esteja marcada o relógio será atualizado com a hora e dia da semana do computador.

Ajustar relógio ao carregar...

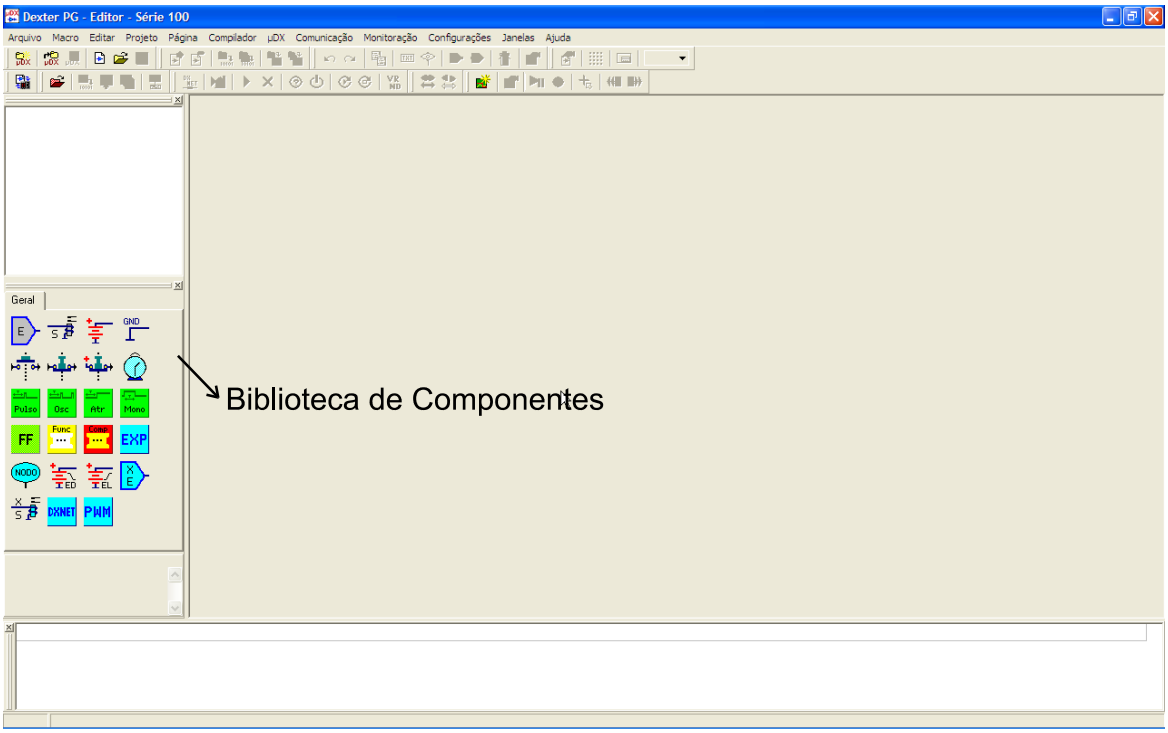
Ajusta o relógio de tempo real da simulação com o valor indicado cada vez que o Simulador é iniciado. A grande diferença em relação à funcionalidade anterior é que, no caso anterior, o relógio simulado somente é ajustado quando o função é chamada, enquanto aqui ele é ajustado sempre que o Simulador é aberto. Isso facilita bastante o teste de programas que tenham dependência de horário. O padrão é estar marcada a opção **Iniciar com data e hora do computador**, ou seja, sempre que o Simulador inicia ele lê a hora e dia de semana do computador e ajusta o relógio simulado. Mas a opção **Iniciar com data e hora indicados acima** é que realmente facilita a depuração de programas. Por exemplo, digamos que determinado programa aplicativo execute uma operação todas as segundas, 10:00. Podemos selecionar para que o relógio do simulador sempre inicie como segunda-feira, 9:59:50. Com isso, bastam 10 segundos de simulação para que a operação seja simulada.


Ajuste de relógio para simulador
Dia
Domingo
Segunda
Terça
Quarta
Quinta
Sexta
Sábado
Hora
12:03:27
Definição do relógio ao carregar programa
Iniciar com data e hora do computador
Iniciar com data e hora indicados acima
Ok
Cancela

Parte

Blocos de Instruções

O Controlador µDX Série 100 (µDX100 ou µDX100+) reconhece e interpreta 23 blocos de instruções diferentes (biblioteca padrão para µDX100 versão 1.0).

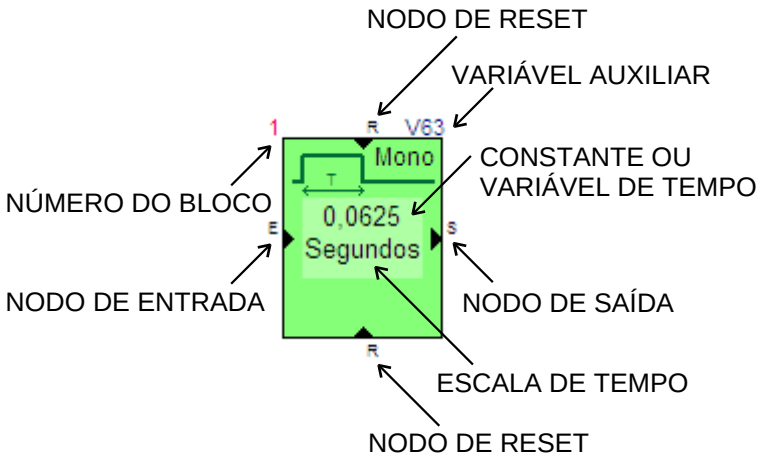


Funcionamento dos Blocos de Tempo

No µDX100 os blocos de temporização (Monoestável, Pulso, Atraso e Oscilador) funcionam da seguinte maneira:

A constante ou a variável especificadas para programar o tempo escolhido são reconhecidas apenas uma vez a cada temporização pelo programa que controla estes blocos. Assim, quando o nodo de entrada de um bloco de monoestável é ligado o programa do µDX reconhece este estado e lê o valor da constante ou o valor atual da variável que indica o tempo programado para o bloco (Tempo).

O valor obtido é então memorizado numa variável auxiliar (Var.Auxiliar) para ser decrementado a cada intervalo de tempo (o intervalo de tempo pode ser programado pelo usuário em horas, minutos, ou segundos).



Bloco de Temporização (Monoestável)

Note que tanto a variável de tempo quanto a variável auxiliar são variáveis do tipo byte e, portanto, podem assumir valores entre 0 e 255. Como o valor 0 (zero) indica que o bloco temporizado chegou ao final da temporização, e o valor 255 que o bloco está pronto para ser disparado, os valores são de 1 a 254. Com isso, conforme a escala de tempo escolhida e o ciclo de execução do programa aplicativo (1/16, 1/32, 1/64 ou 1/256 s) pode-se programar os seguintes tempos:

Ciclo de Execução do programa aplicativo = 1/16s (62,5ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	62,5 ms	15,875 s
Minutos	0,25 min = 15 s	63,5 min.= 1 h 3 min 30 s
Horas	0,067 h = 4 min	16,933 h = 16 h 56 min

Ciclo de Execução do programa aplicativo = 1/32s (31,25ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	31,25 ms	7,9375 s
Minutos	0,125 min = 7,5 s	31,75 min.= 31 min 45 s
Horas	0,033 h = 2 min	8,467 h = 8 h 28 min

Ciclo de Execução do programa aplicativo = 1/64s (15,625ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	15,625 ms	3,9688 s
Minutos	0,0625min = 3,75s	15,875 min.= 15 min 52,5 s
Horas	0,017 h = 1 min	4,233 h = 4 h 14 min

Ciclo de Execução do programa aplicativo = 1/256s (3,90625ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	3,90625 ms	0,9922 s
Minutos	0,0156min = 0,94s	3,9688 min.= 3 min 58,1 s
Horas	0,0042 h = 15,08 s	1,0636 h = 1 h 3 min 49 s

Quando o valor nesta variável auxiliar chegar a zero o programa do µDX vai executar a segunda parte da função do bloco. No caso do monoestável isto significa desligar o nodo de saída e para os outros tipos (como o blocos de atraso) seria o momento de ligar o nodo de saída.

Depois o programa do µDX manterá o valor na variável auxiliar em zero até que o nodo de entrada seja desligado e novamente ligado ou o nodo de reset seja acionado, causando um zeramento do bloco. Este zeramento significa atribuir valor 255 à variável auxiliar, o que indica ao µDX100 que este bloco está zerado e pronto para nova temporização.

Como o µDX não dispõe de espaço de RAM interna para variáveis auxiliares, estas são obtidas pela utilização das variáveis de cálculo disponíveis. Para reduzir a possibilidade de conflito entre as variáveis auxiliares usadas pelos blocos temporizadores e as variáveis utilizadas no programa aplicativo a alocação de variáveis auxiliares é feita em ordem decrescente (similar a pilha usada em processadores), isto é, da variável v15 até a variável v0 (ou, no caso de µDX100+, de variável v63 até variável v0).

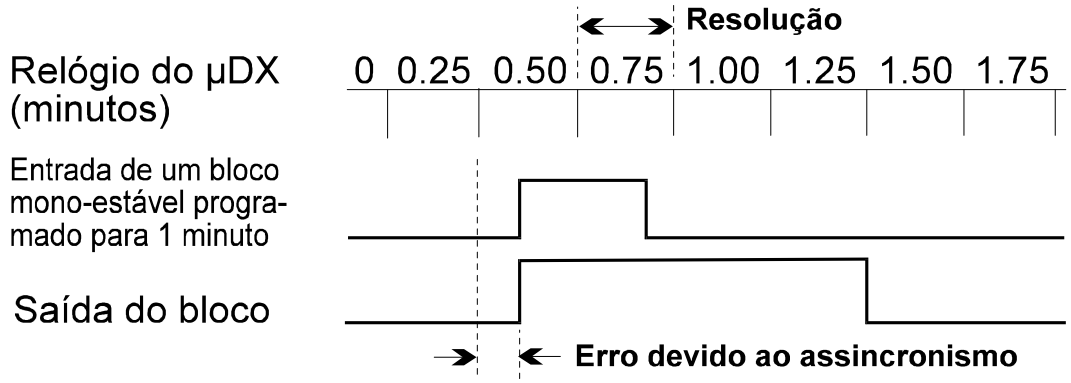
A variável escolhida aparece no canto superior direito do bloco sempre que é efetuada a pré-compilação do programa aplicativo. Para cada bloco de temporização inserido na área de programação a variável escolhida nos outros blocos pode mudar. Este efeito ocorre porque o µDX100 é que obtém qual variável ele vai empregar como auxiliar conforme a ordem dos blocos ao longo de todo o programa.

Note que qualquer modificação no valor de uma variável que se usa para programar o tempo de um destes blocos não será notada senão quando ele for ativado. Por outro lado, se o programa feito pelo usuário interferir no valor da variável utilizada como auxiliar, isto vai resultar em alteração do período de tempo de atividade do bloco (por exemplo, empregando-se um bloco de função para mudar o valor da variável auxiliar quando ocorrer determinada situação). Este recurso pode ser empregado em programas mais avançados.

ATENÇÃO: Caso a duração de ciclo especificada na página de programação (1/16, 1/32, 1/64, ou 1/256s) seja diferente de 1/16s as temporizações mudarão suas escalas de acordo. Assim para 1/32s um intervalo especificado para 2,5 segundos passará em 1,25 segundos e para 1/64s passará em apenas 0,625 segundos (No caso de µDX com firmware igual ou superior à versão 4.2 existe a opção de 1/256s. Neste caso o intervalo de 2,5 segundos teria duração de 0,15625 segundos, desde que o programa seja pequeno para não comprometer a velocidade de execução).

ATENÇÃO: Para cada faixa de tempo escolhida existe uma resolução para a duração ajustada. Assim, por exemplo, a faixa de horas tem resolução de 4 minutos (0.066 horas), o que quer dizer que a duração programada é sempre um intervalo múltiplo de 4 minutos. Além disso, esta resolução torna o intervalo total programado dependente do momento em que o temporizador é

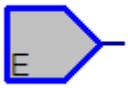
disparado. Isto ocorre porque para cada faixa os sinais internos que fazem a contagem do tempo programado ocorrem em momentos fixos. Veja a figura:



Assim, para temporizadores selecionados para a faixa de segundos existe uma incerteza de 0,06s. Já para temporizadores selecionados para a faixa de minutos esta incerteza é de 15s. Por fim, para temporizadores selecionados para a faixa de horas a incerteza é de 4 minutos. Estes valores são para configuração de 1/16s. Caso seja selecionada uma configuração de 1/32s todas as incertezas acima são divididas por 2. Para as outras faixas de operação do µDX (1/64 ou 1/256s) vale a regra de divisão correspondente. Entretanto, em velocidades muito altas de operação, programas grandes e temporizações curtas, podem ocorrer atrasos adicionais devido ao processamento do programa.

GERAL - Entrada Digital

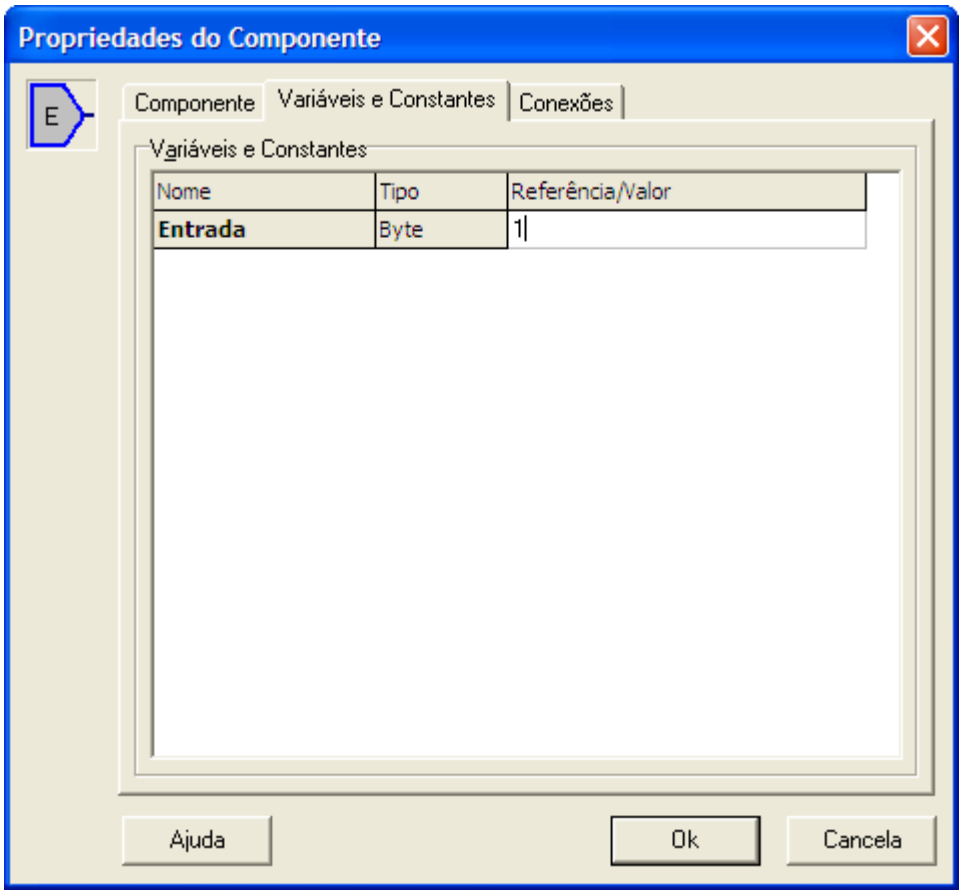
Este bloco acessa as entradas digitais do controlador programável µDX100. Assim como indicado na tampa superior do µDX100, cada entrada tem uma numeração própria: E1, E2, E3 e E4. O bloco permite edição para especificar qual entrada está sendo usada. A conexão de saída deste bloco, como de todos os blocos do µDX100, é um nodo, ou seja, uma variável binária (apenas dois estados, ligado ou desligado).



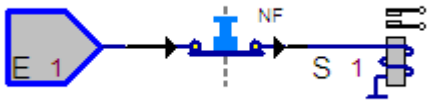
Internamente ao µDX100, as entradas são associadas a nodos (0 a 3) e, a cada ciclo, o programa do µDX verifica o estado de cada entrada e liga ou não o nodo correspondente. Além disso, as entradas E1, E2 e E3 podem servir para a leitura da largura de pulso quando se emprega o bloco de instrução **PWM** (ver **GERAL - PWM**). Como as entradas do µDX100 estão associadas a números de nodos (n0 a n3) este bloco não é contabilizado no total de blocos do programa aplicativo. Portanto, não há limite para seu uso. Na verdade ele equivale a um bloco de nodo acessando ao nodo absoluto n0 a n3.

ATENÇÃO: As entradas do µDX100 não são isoladas galvanicamente (as entradas da Expansão são isoladas). Portanto, não conecte os fios da energia elétrica domiciliar diretamente em qualquer das entradas do µDX. Utilize para isso o módulo de Opto-Acoplador ou um circuito externo com isolador ótico (veja Entradas e Saídas).

Ao editar o bloco (aponte com o mouse para o bloco e pressione a tecla direita do mouse, ou pressione a tecla de espaço no teclado do computador) surge uma tela para inserção da entrada a ser usada. Digite um valor entre 1 e 4:



O exemplo a seguir liga a saída S1 do µDX100 quando a entrada E1 for energizada. Note que foi usada uma chave NF (normal fechada) para isolar a entrada digital da saída digital. Não é permitido ligar diretamente entradas e saídas do controlador µDX100. Como as entradas e saídas, na verdade, são acessos a nodos absolutos (n0 a n3 para as entradas; n4 a n7 para as saídas) a conexão direta entre estes blocos equivale a especificar dois nodos distintos para a mesma conexão. Isso gera um erro de compilação.



Exemplo de Programa Aplicativo: **Bloco_entrada.d1g**

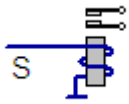
Veja também:
GERAL - Saída Digital

GERAL - Saída Digital

As quatro saídas do µDX100 - S1, S2, S3 e S4 - são feitas utilizando-se relés, cada um com um contato reversor, isto é, existe neles uma lâmina móvel que permite passar a corrente elétrica quando está encostando em um dos dois contatos elétricos montados de cada lado. Assim, quando o relé está desligado a lâmina fica encostando em um deles (Normal Fechado - NF) e afastada do outro (Normal Aberto - NA). Quando o relé é ligado a conexão se reverte, ficando o

primeiro contato em aberto e o segundo fechado.

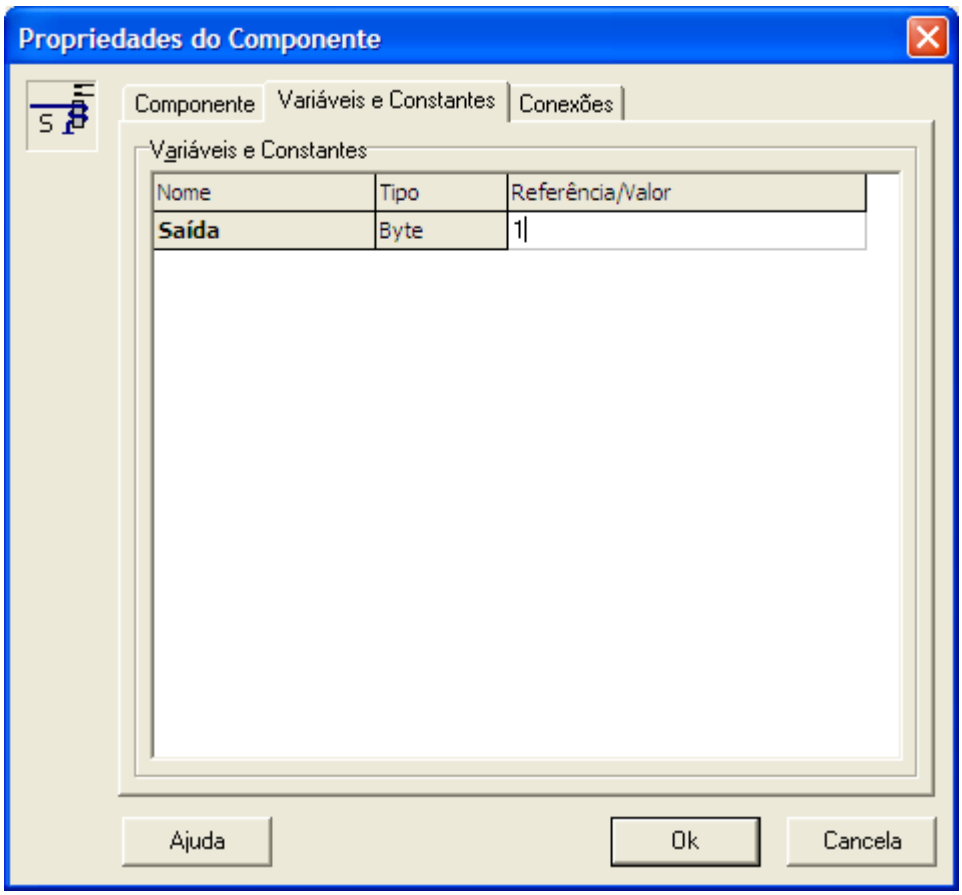
Como os dois contatos e a lâmina central estão disponíveis para conexões externas (através do conector lateral do µDX100) ambos podem ser empregados para que se tenha corrente elétrica quando o relé está desligado (usando o contato NF) ou quando está ligado (usando o contato NA).



Como as entradas, as quatro saídas também são, internamente, consideradas como nodos (n4 a n7). Logo, se o nodo correspondente a uma das saídas for ligado por algum bloco de instrução (ou mesmo forçado como ligado via DXNET) então o relé desta saída será energizado. Este bloco não é contabilizado no total de nodos do programa aplicativo, já que na verdade ele apenas atribui um nodo específico a conexão (n4 a n7).

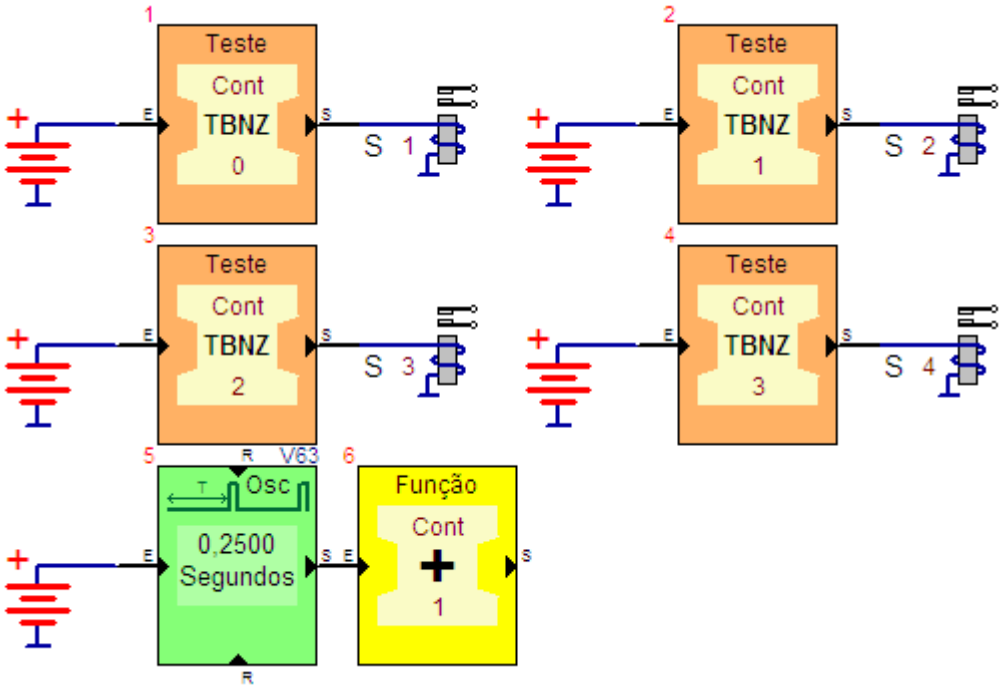
ATENÇÃO: Como cada relé proporciona uma isolação galvânica suficiente, você pode conectar em seus contatos os fios da energia elétrica domiciliar, observando apenas o limite de corrente máximo de 10 Ampéres. Tome sempre muito cuidado nas conexões devido à proximidade entre os contatos do conector.

Ao editar o bloco (aponte com o mouse para o bloco e pressione a tecla direita do mouse, ou pressione a tecla de espaço no teclado do computador) surge uma tela para inserção da saída a ser usada. Digite um valor entre 1 e 4:



O exemplo a seguir liga as saídas S1 a S4 do controlador µDX100 em uma contagem binária. Como a variável **Cont** é incrementada a cada 250ms e é testado os 4 primeiros bits desta variável para acionamento das quatro saídas do controlador, as saídas apresentam uma contagem binária

de 0 a 15 (0000b a 1111b):

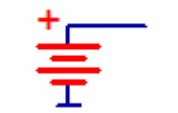


Exemplo de Programa Aplicativo: **Bloco_saida.d1g**

Veja também:
GERAL - Entrada Digital

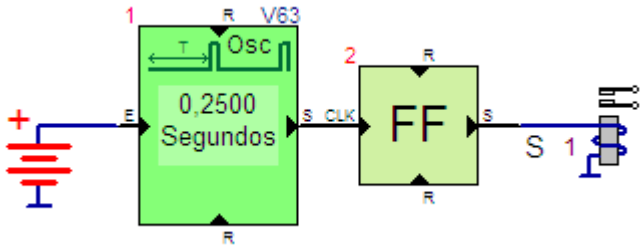
GERAL - Energia

Este bloco serve para indicar quais nodos devem ser forçadamente energizados (ligados). Este bloco é interpretado pelo µDX100 como sendo um forçamento do nodo para o estado ligado constantemente. Internamente, este nodo será o nodo n63, que é mantido sempre ligado.



Assim, sempre que um bloco de instrução ou uma ligação que una um ou mais blocos de instrução precisar ficar continuamente no estado ligado, basta conectar a este bloco de energia. Note que este bloco equivale a um bloco nodo acessando o nodo absoluto n63. Este bloco não é contabilizado no total de nodos do programa aplicativo, já que na verdade ele apenas atribui um nodo específico a conexão (n63).

Abaixo temos um exemplo de uso deste bloco para ativar constantemente o bloco de Oscilador, que gera uma onda quadrada (via FF) na saída S1 do controlador µDX100.



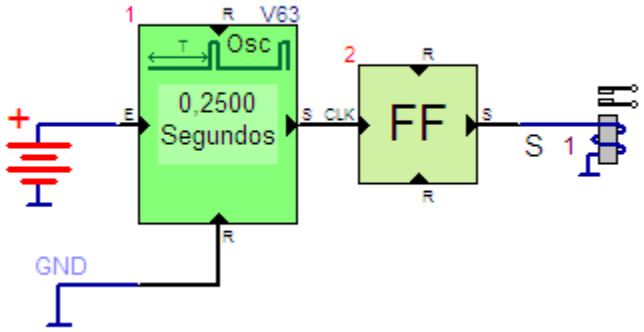
Exemplo de Programa Aplicativo: **Bloco_energia.d1g**

GERAL - Terra

Este bloco serve para indicar quais nodos devem ser forçadamente desenergizados (desligados). Este bloco é interpretado pelo µDX100 como sendo um forçamento do nodo para o estado desligado constantemente. Internamente, este nodo será o nodo número 62, que é mantido sempre desligado.

Assim, sempre que um bloco de instrução ou uma ligação que una um ou mais blocos de instrução precisar ficar continuamente no estado desligado, basta conectar a este bloco de energia. Note que este bloco equivale a um bloco nodo acessando o nodo absoluto n62. Este bloco não é contabilizado no total de nodos do programa aplicativo, já que na verdade ele apenas atribui um nodo específico a conexão (n62).

Abaixo temos um exemplo de uso deste bloco para desativar constantemente o nodo de reset do bloco de Oscilador do exemplo anterior, que gera uma onda quadrada (via FF) na saída S1 do µDX100. Note que poderíamos, simplesmente, deixar esta entrada de reset sem nenhuma conexão (qualquer nodo de entrada sem nenhuma conexão é mantido no estado zero).

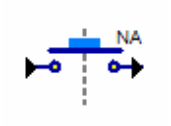


Exemplo de Programa Aplicativo: **Bloco_GND.d1g**

GERAL - Chave NA

A chave NA (Normalmente Aberta) é um bloco cuja função está descrita pela própria representação gráfica: como um interruptor de corrente elétrica o botão que aparece no centro tem dois contatos por onde "passará" a corrente em determinadas condições.

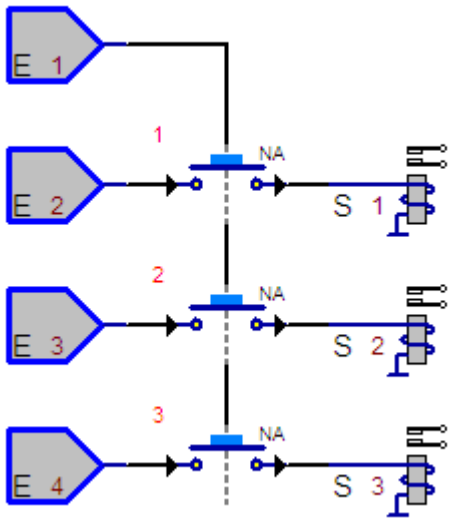
A chave Normalmente Aberta (NA) precisa que o nodo de controle (linha pontilhada) seja ativado para que ela permita passar o estado do nodo de entrada para o nodo de saída. Note que o nodo de controle possui dois pontos de conexão: um na parte superior e outro na parte inferior do bloco.



Um exemplo de uso de chave NA é o interruptor de campainha de residências. Ao acioná-lo ele permite a passagem de corrente elétrica, acionando a campainha.

Note que a chave NA, ao contrário de seu similar físico (interruptor elétrico), permite a transmissão de sinal apenas do nodo de entrada (nodo à esquerda) para o nodo de saída (nodo à direita), como indicado pelas setas. Caso outro bloco ative o nodo de saída de uma chave NA, o nodo de entrada da mesma não será acionado, mesmo que a chave esteja fechada (nodo de controle ligado).

No exemplo a seguir, quando a entrada E1 do µDX100 é acionada, permite que os sinais presentes nas entradas E2, E3 e E4 do mesmo módulo sejam replicados nas saídas S1, S2, e S3, respectivamente. Quando E1 está desativado as chaves NA abrem e nenhuma saída é ativada, independentemente do estado de E2, E3 e E4. Note que E1 controla as três chaves NA, já que os nodos de controle delas estão interligados.



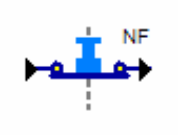
Exemplo de Programa Aplicativo: **Bloco_chaveNA.d1g**

Veja também:
GERAL - Chave NF
GERAL - Chave Inversora

GERAL - Chave NF

A chave NF (Normalmente Fechada) é um bloco cuja função está descrita pela própria representação gráfica: como um interruptor de corrente elétrica o botão que aparece no centro tem dois contatos por onde "passará" a corrente em determinadas condições.

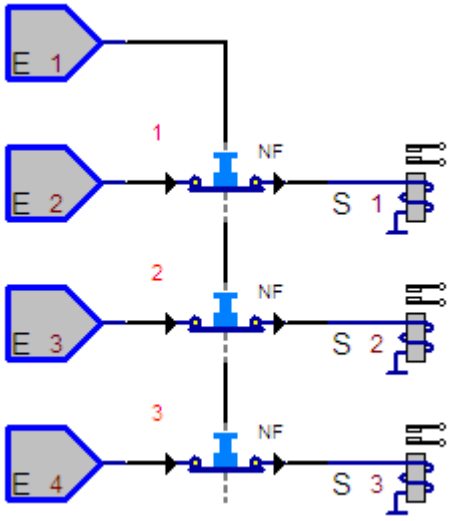
A chave Normalmente Fechada (NF) precisa que o nodo de controle (linha pontilhada) seja desativado para que ela permita passar o estado do nodo de entrada para o nodo de saída. Note que o nodo de controle possui dois pontos de conexão: um na parte superior e outro na parte inferior do bloco.



Um exemplo de chave NF usada comumente é a chave de iluminação interna de refrigeradores. Quando pressionada a lâmpada permanece desligada (porta fechada), e ao desacionar a chave (porta aberta) a lâmpada acende.

Note que a chave NF, ao contrário de seu similar físico (interruptor elétrico), permite a transmissão de sinal apenas do nodo de entrada (nodo à esquerda) para o nodo de saída (nodo à direita), como indicado pelas setas. Caso outro bloco ative o nodo de saída de uma chave NF, o nodo de entrada da mesma não será acionado, mesmo que a chave esteja fechada (nodo de controle desligado).

No exemplo a seguir, quando a entrada E1 do controlador µDX100 é desacionada, permite que os sinais presentes nas entradas E2, E3 e E4 do mesmo módulo sejam replicados nas saídas S1, S2, e S3, respectivamente. Quando E1 está ativado as chaves NF abrem e nenhuma saída é ativada, independentemente do estado de E2, E3 e E4. Note que E1 controla as três chaves NF, já que os nodos de controle delas estão interligados.

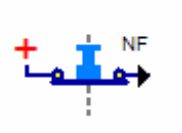


Exemplo de Programa Aplicativo: **Bloco_chaveNF.d1g**

Veja também:
GERAL - Chave NA
GERAL - Chave Inversora

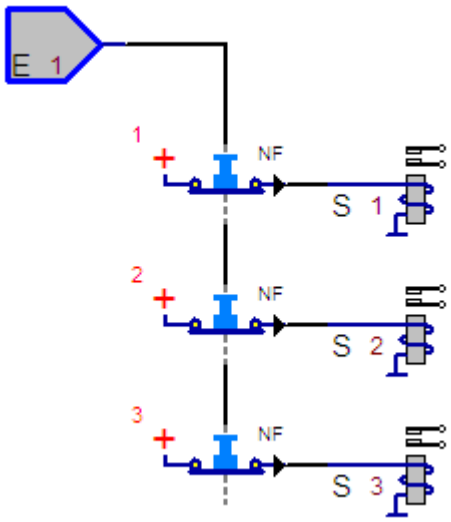
GERAL - Chave Inversora

A chave Inversora é um bloco cuja função está descrita pela própria representação gráfica: como um interruptor de corrente elétrica o botão que aparece no centro tem dois contatos por onde "passará" a corrente em determinadas condições. Na verdade, se trata de uma chave NF (Normalmente Fechada) com o nodo de entrada constantemente ligado. Com isso, caso se ative o nodo de controle a chave abre e o nodo de saída é desligado. Já se for desligado o nodo de controle a chave fecha e o nodo de saída será ativado. Com isso, o estado do nodo de saída é invertido em relação ao estado do nodo de controle.



Note que o nodo de controle possui dois pontos de conexão: um na parte superior e outro na parte inferior do bloco.

No exemplo a seguir, quando a entrada E1 do controlador µDX100 é desacionada, ativa as três saídas S1, S2 e S3 do mesmo módulo. Já quando esta entrada é acionada desativa as três saídas. Note que E1 controla as três chaves NF, já que os nodos de controle delas estão interligados.

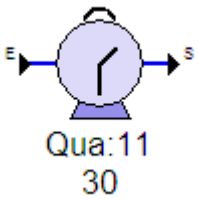


Exemplo de Programa Aplicativo: **Bloco_chaveInv.d1g**

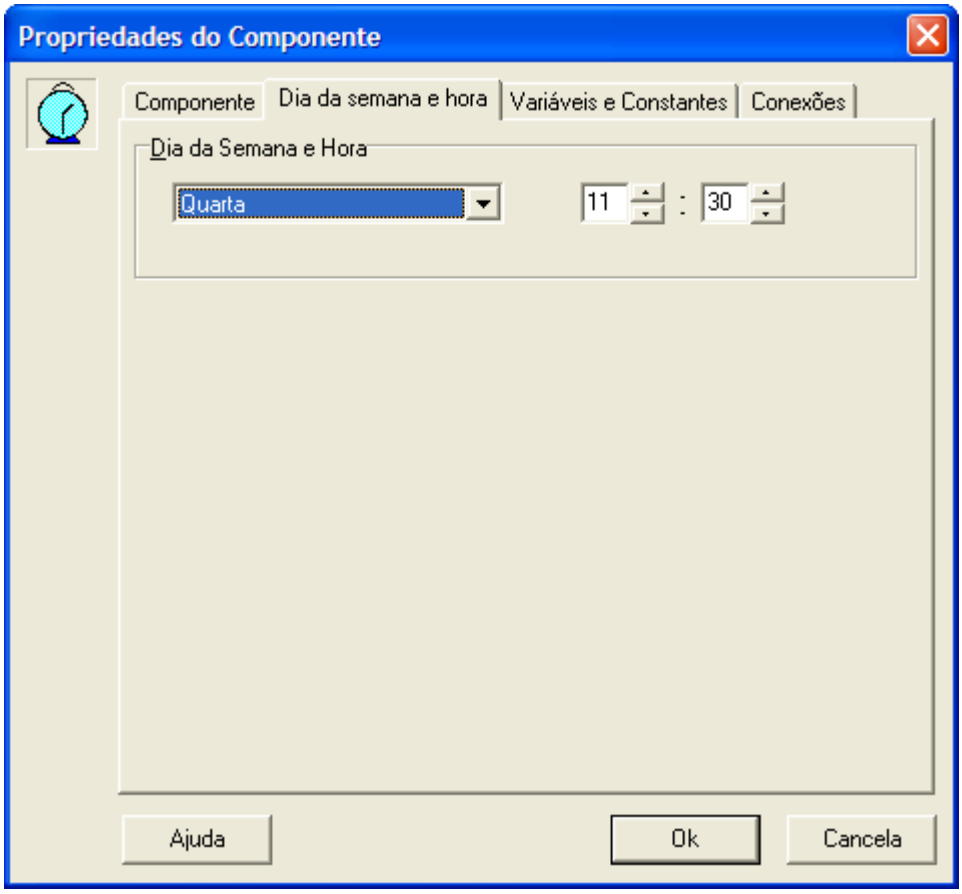
Veja também:
GERAL - Chave NA
GERAL - Chave NF

GERAL - Relógio

Este bloco liga o nodo de saída (S) quando o horário do relógio de tempo real do controlador programável µDX100 coincidir com o horário especificado no bloco. No desenho abaixo, por exemplo, isso acontece toda quarta-feira, às 11 horas e 30 minutos. Para que o nodo de saída (S) gere o pulso é necessário que o nodo de entrada (E) esteja energizado durante o horário programado.



Note que o bloco permite tanto especificar um horário específico, como usar uma variável para especificar este horário. No caso de horários específicos, que é o caso mais comum, podemos selecionar o dia da semana (entre Segunda, Terça, Quarta, Quinta, Sexta, Sábado, ou Qualquer), a hora (entre 0 e 23), e minuto (entre 0 e 59).



Caso seja especificado dia da semana qualquer, significa que todos os dias no horário especificado o bloco será acionado (desde que o nodo de entrada esteja energizado).

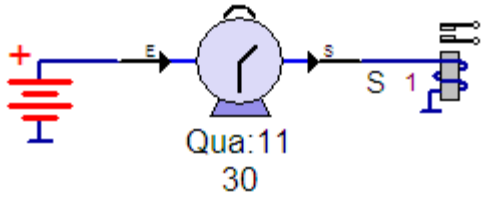
Esta instrução utiliza o relógio interno do µDX para comparar com o dia, as horas e os minutos programados e, quando coincidirem, acionar o nodo de saída durante um minuto (já que os segundos não são comparados o horário coincide por um minuto). Mesmo que o relógio interno tenha correspondência, se o nodo de entrada não estiver ativo o nodo de saída permanecerá desligado.

O horário de acionamento pode ser programado utilizando-se variáveis ou constantes (combinadas livremente). Assim pode-se escolher uma variável para os minutos e deixar o dia e a hora indicados por constante.

Na figura deste bloco o dia e a hora aparecem na parte inferior, abaixo do desenho do relógio, na primeira linha. Os minutos aparecem na segunda linha. A posição dos ponteiros do relógio nada tem a ver com os valor especificados, servindo apenas para composição artística do desenho.

Quando for escolhida uma constante para o dia e hora é possível assinalar uma condição para que o relógio seja atuado em qualquer dia da semana e num determinado horário. Para isso deve-se escolher a opção de **Qualquer** dentre os dias da semana (abaixo do relógio aparecerá **XX**).

O programa abaixo liga a saída todas as quartas-feiras, às 11:30:



Exemplo de Programa Aplicativo: **Bloco_relogio.d1g**

Quando utilizada uma variável para minutos basta especificar nesta variável os minutos desejados multiplicados por quatro (isso porque os dois bits menos significativos não são usados):

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
m	m	m	m	m	m	x	x

Quando utilizada uma variável para o dia e hora deve-se também adequar o valor dela com o formato do byte de dia e hora do relógio do µDX100:

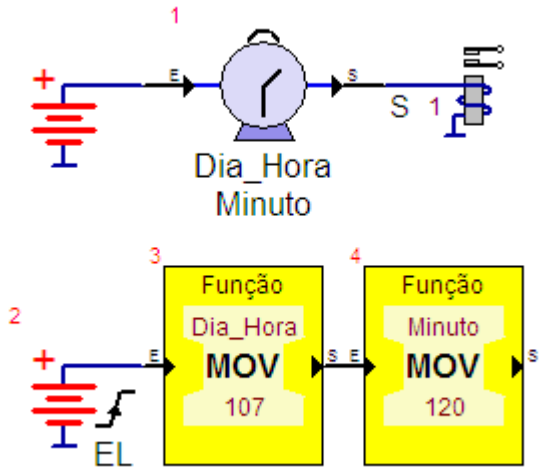
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
d	d	d	h	h	h	h	h

Onde: **d** - indica o dia (000=domingo, 001=segunda,... e 111 para todos os dias)
h - indica as horas (0 a 23 em decimal, ou 00000 a 10111 em binário)
m - indica os minutos (0 a 59 em decimal, ou 000000 a 111011 em binário)
x - qualquer valor, não é usado.

Por exemplo, para especificar o mesmo horário de 11:30 todas as quartas-feiras via variáveis teríamos os seguintes valores:

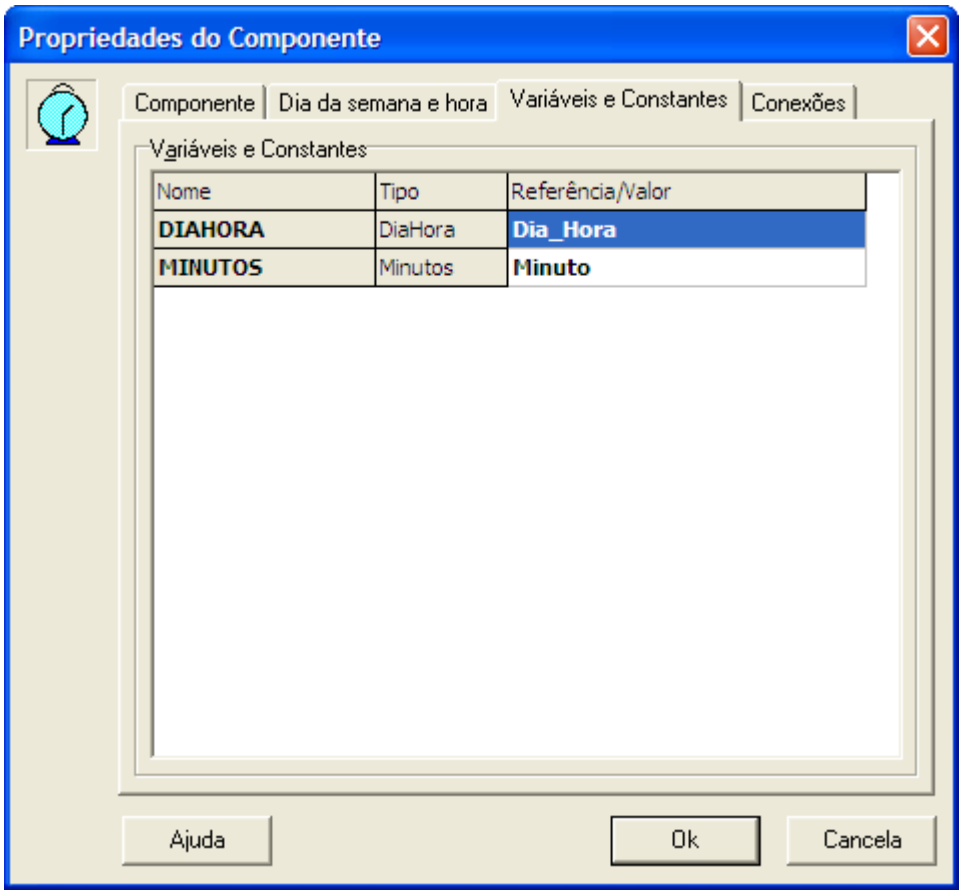
Minutos: 30 x 4 = 120
Dia/Hora: Quarta (011) + 11h (01011b) = 01101011b = 107

Logo, teríamos o programa a seguir:



Exemplo de Programa Aplicativo: **Bloco_relogio2.d1g**

A seleção das variáveis é feita na tela de edição do bloco, aba **Variáveis e Constantes**:



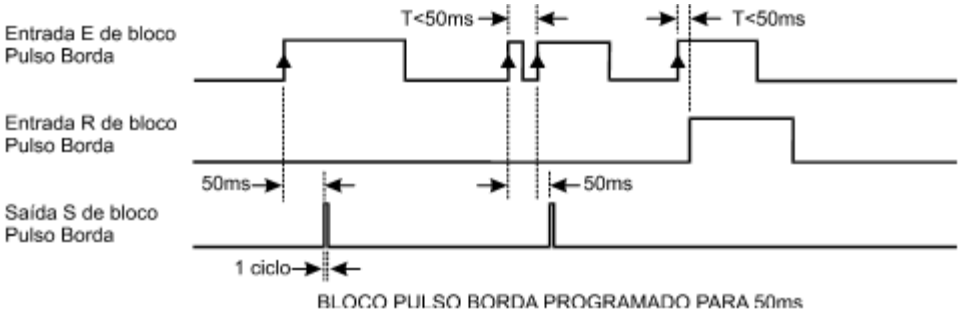
Note que, evidentemente, o relógio de tempo real do controlador µDX100 deve estar inicializado com o horário e dia da semana corretos para o funcionamento adequado deste bloco. Para acertar o relógio do µDX100 basta acessar o comando de **Acertar Relógio...** existente no menu pop-down **µDX** do Compilador PG.

GERAL - Pulso

Este bloco retarda a energização do nodo de saída S durante o tempo programado, quando detecta energização do nodo de entrada E, e mantêm a saída ligada somente durante um ciclo do CLP. Note que este bloco irá gerar o pulso na saída S mesmo que a entrada E retorne a zero antes de decorrido o tempo programado. Para um novo disparo é preciso que o nodo de entrada (E) retorne a zero e volte a subir.

Os nodos de Reset (R) estão internamente interligados e permitem desativar o bloco imediatamente. Note que qualquer sinal ligado a um dos nodos R está automaticamente ligado ao outro nodo R, já que se trata de dois nodos ligados internamente pelo bloco.

Abaixo temos uma representação gráfica do comportamento do bloco de Pulso Borda:



BLOCO PULSO BORDA PROGRAMADO PARA 50ms

Para especificar o tempo do bloco Pulso pode-se utilizar uma constante ou uma variável. As faixas de duração disponíveis são: Minutos e Segundos. Os tempos variam conforme a

velocidade de execução do programa e a faixa escolhida:

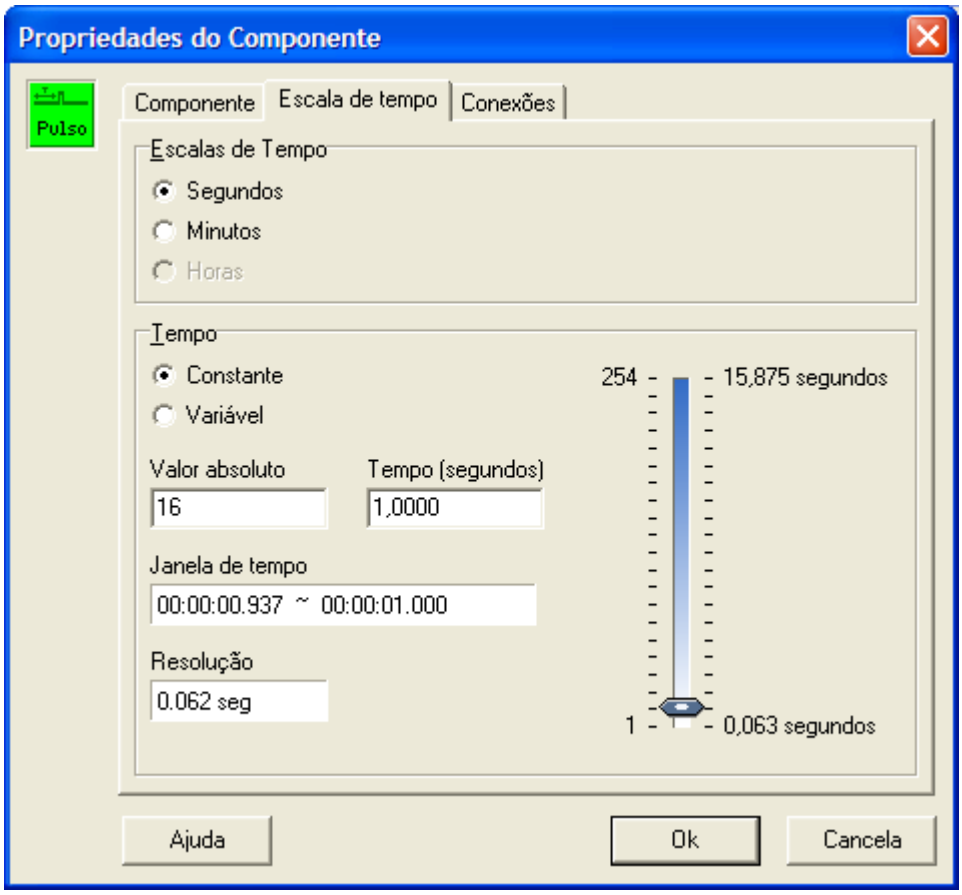
Ciclo de Execução de 1/16s (62,5ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	62,5ms	15,875s
Minutos	0,25min = 15s	63,5min = 1h 3min 30s

Ciclo de Execução de 1/32s (31,25ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	31,25ms	7,9375s
Minutos	0,125min = 7,5s	31,75min = 31min 45s

Ciclo de Execução de 1/64s (15,625ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	15,625ms	3,9688s
Minutos	0,063min = 3,75s	15,875min = 15min 52,5s

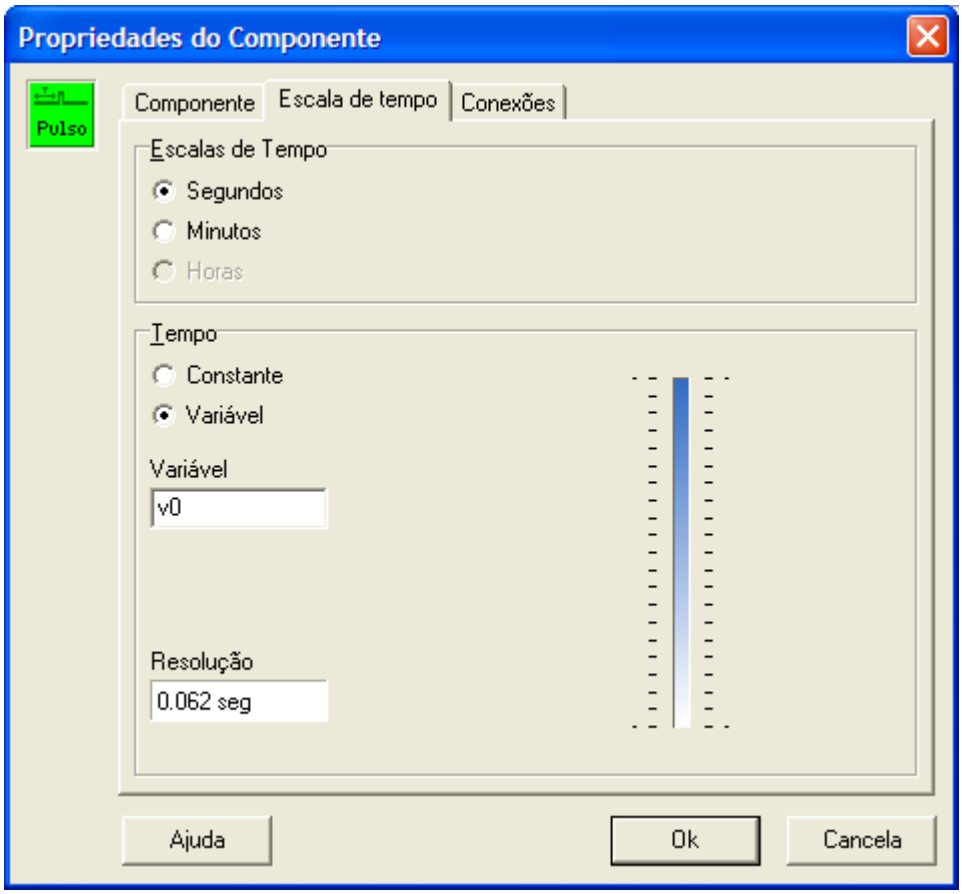
Ciclo de Execução de 1/256s (3,9063ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	3,9063ms	0,9922s
Minutos	0,016min = 0,94s	3,9688min = 3min 58,1s

A seguir temos a tela de edição deste bloco, no caso usando uma valor constante de 1 segundo para o tempo do bloco Pulso:

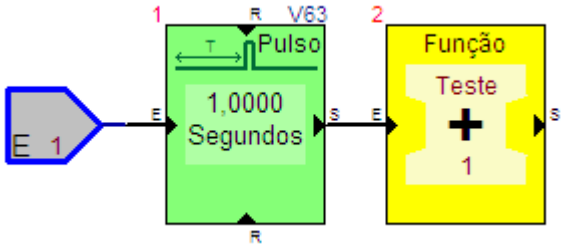


A barra de seleção do tempo pode ser arrastada com o mouse ou usar as setas do teclado do computador. Também é possível digitar o tempo desejado no quadro Tempo. O tempo digitado será arredondado para o valor mais próximo ao pressionar a tecla **[Ok]**.

Por fim, em vez de um valor constante para o tempo deste bloco, pode-se escolher uma variável que determinará o tempo do bloco Pulso:



ATENÇÃO: No caso de blocos de temporização utilizando ciclo de execução de 1/256s é necessário que o tempo de ciclo do programa aplicativo seja menor ou igual a 1/256s (3,9ms), ou o bloco de temporização sofrerá um atraso na temporização, devido ao tempo de ciclo do programa ser superior à resolução da temporização.



Exemplo de Programa Aplicativo: **Bloco_Pulso.d1g**

Acima temos um exemplo de uso do bloco de temporização Pulso. O programa gera um pulso de apenas um ciclo a cada energização da entrada E1 do µDX100. Este pulso é usado para incrementar a variável **Teste** a cada nova energização de E1. A entrada E1 somente incrementa a variável var1 a cada segundo, independentemente do número de energizações em E1. Ou seja, o

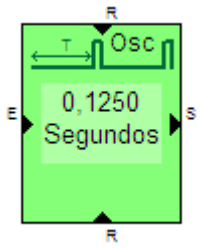
µDX100 irá ficar sensível a entrada E1 apenas para um pulso por segundo, sendo que os demais serão desprezados.

Importante: como todos os blocos de temporização, o bloco Pulso usa uma variável auxiliar para efetuar a contagem de tempo. Esta variável é alocada automaticamente a partir da última variável disponível no controlador (v15 no caso de µDX100, v63 no caso de µDX100+) e aparece no canto superior direito do bloco quando é efetuada a pré-compilação. Cuidado para que a lista de variáveis usadas no programa aplicativo (lista crescente) não encontre a lista de variáveis auxiliares para blocos temporizadores (lista decrescente).

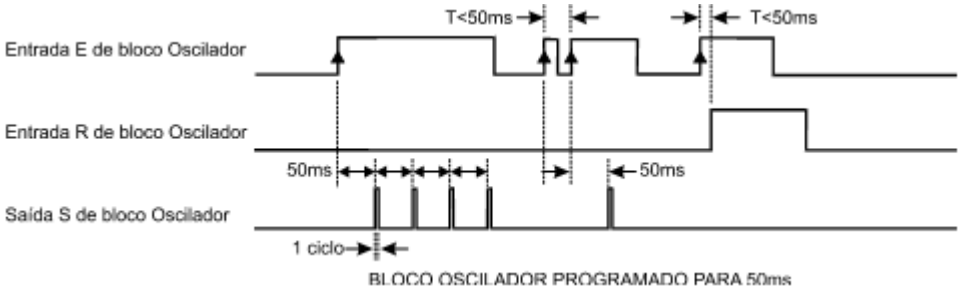
GERAL - Oscilador

Este bloco gera pulsos no nodo de Saída (S) com largura de um ciclo de execução do programa aplicativo, a cada tempo programado pelo operando Tempo. Para habilitá-lo é necessário energizar o nodo de Entrada (E).

Os nodos de Reset (R) estão internamente interligados e permitem desativar o bloco imediatamente. Note que qualquer sinal ligado a um dos nodos R está automaticamente ligado ao outro nodo R, já que se trata de dois nodos ligados internamente pelo bloco.



Abaixo temos uma representação gráfica do comportamento do bloco de Oscilador:



BLOCO OSCILADOR PROGRAMADO PARA 50ms

Para especificar o tempo do bloco Oscilador (que irá determinar o período de oscilação) pode-se utilizar uma constante ou uma variável. As faixas de duração disponíveis são: Minutos e Segundos. Os tempos variam conforme a velocidade de execução do programa e a faixa escolhida:

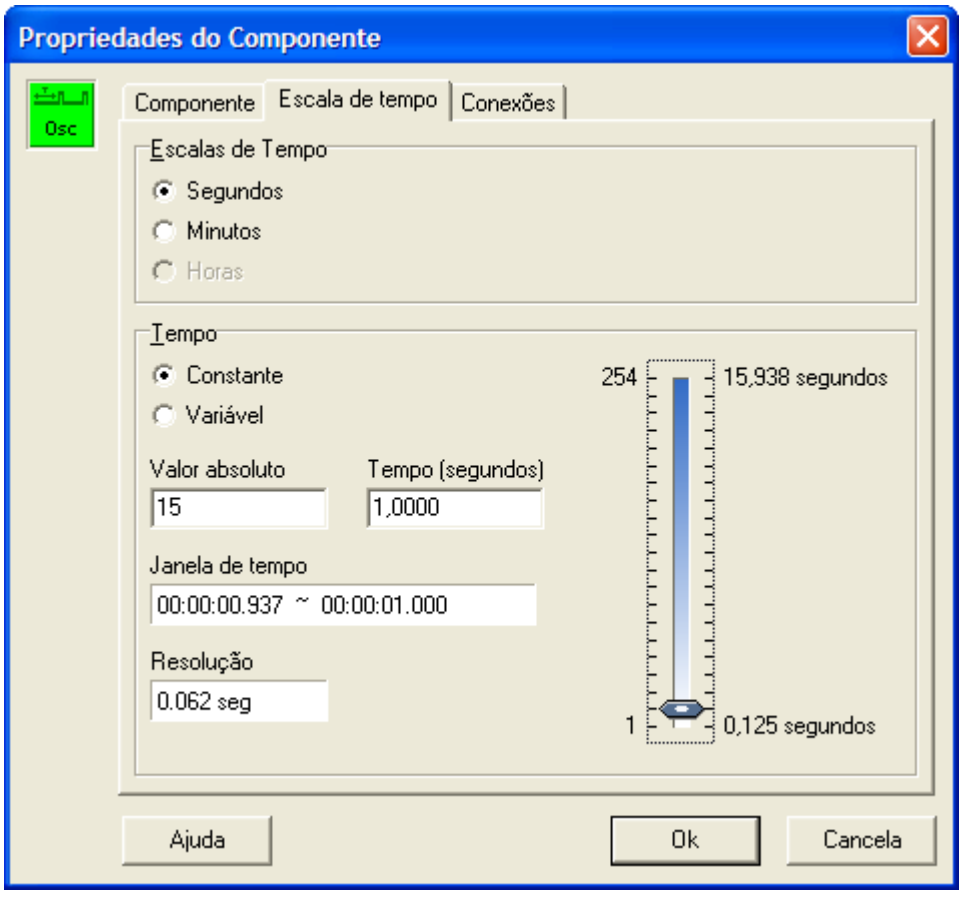
Ciclo de Execução de 1/16s (62,5ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	125ms	15,9375s
Minutos	0,25min = 15s	63,5min = 1h 3min 30s

Ciclo de Execução de 1/32s (31,25ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	62,5ms	7,9688s
Minutos	0,125min = 7,5s	31,75min = 31min 45s

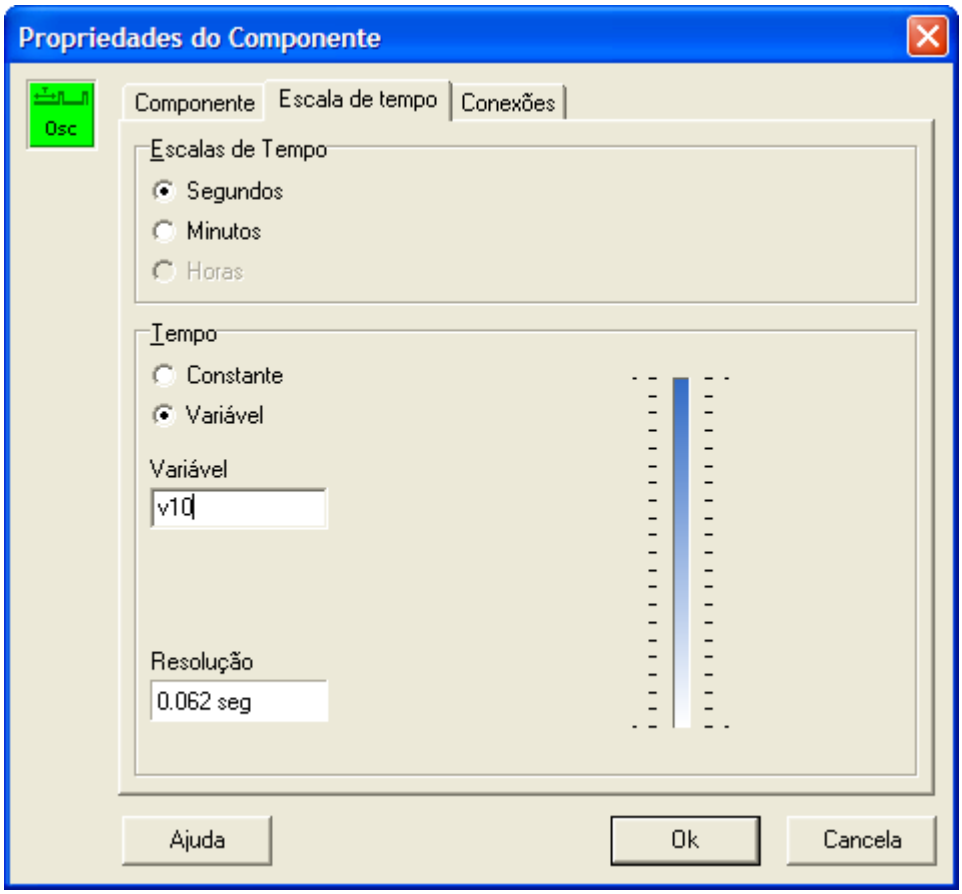
Ciclo de Execução de 1/64s (15,625ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	31,25ms	3,9844s
Minutos	0,063min = 3,75s	15,875min = 15min 52,5s

Ciclo de Execução de 1/256s (3,9063ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	7,8126ms	0,9961s
Minutos	0,016min = 0,94s	3,9688min = 3min 58,1s

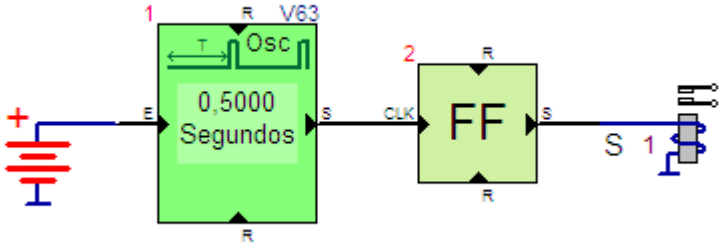
A seguir temos a tela de edição deste bloco, no caso usando uma valor constante de 1 segundo para o tempo do bloco Oscilador:



A barra de seleção do tempo pode ser arrastada com o mouse ou usar as setas do teclado do computador. Também é possível digitar o tempo desejado no quadro Tempo. O tempo digitado será arredondado para o valor mais próximo ao pressionar a tecla **[Ok]**. Por fim, em vez de um valor constante para o tempo deste bloco, pode-se escolher uma variável que determinará o tempo do bloco Oscilador:



ATENÇÃO: No caso de blocos de temporização utilizando ciclo de execução de 1/256s é necessário que o tempo de ciclo do programa aplicativo seja menor ou igual a 1/256s (3,9ms), ou o bloco de temporização sofrerá um atraso na temporização, devido ao tempo de ciclo do programa ser superior à resolução da temporização.



Exemplo de Programa Aplicativo: **Bloco_Oscilador.d1g**

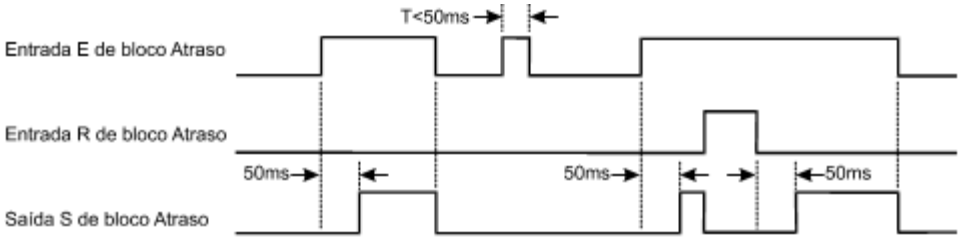
Acima temos um exemplo de uso do bloco de temporização Oscilador. O bloco Oscilador gera pulsos constantemente na saída S (já que a entrada E do bloco está sempre ativada). Estes pulsos trocam de estado o bloco de Flip-flop. A cada pulso a saída S do flip-flop (FF) é invertida. Com isso, surge uma onda quadrada de 1Hz na saída S1 do controlador µDX100.

Importante: como todos os blocos de temporização, o bloco Oscilador usa uma variável auxiliar para efetuar a contagem de tempo. Esta variável é alocada automaticamente a partir da última variável disponível no controlador (v15 no caso de µDX100, v63 no caso de µDX100+) e aparece no canto superior direito do bloco quando é efetuada a pré-compilação. Cuidado para que a lista de variáveis usadas no programa aplicativo (lista crescente) não encontre a lista de variáveis auxiliares para blocos temporizadores (lista decrescente).

GERAL - Atraso

Este bloco gera um atraso entre a energização do nodo de Entrada (E) e a energização do nodo de Saída (S). Caso o nodo de entrada retorne a zero antes do tempo de atraso programado o nodo de saída não é acionado. Quando o nodo de entrada é desenergizado a saída imediatamente também é desligada. Os nodos de Reset (R) estão internamente interligados e permitem desativar o bloco imediatamente. Note que qualquer sinal ligado a um dos nodos R está automaticamente ligado ao outro nodo R, já que se trata de dois nodos ligados internamente pelo bloco.

Abaixo temos uma representação gráfica do comportamento do bloco de Atraso:



BLOCO ATRASO PROGRAMADO PARA 50ms

Para especificar o tempo do bloco Atraso pode-se utilizar uma constante ou uma variável. As faixas de duração disponíveis são: Horas, Minutos e Segundos. Os tempos variam conforme a velocidade de execução do programa e a faixa escolhida:

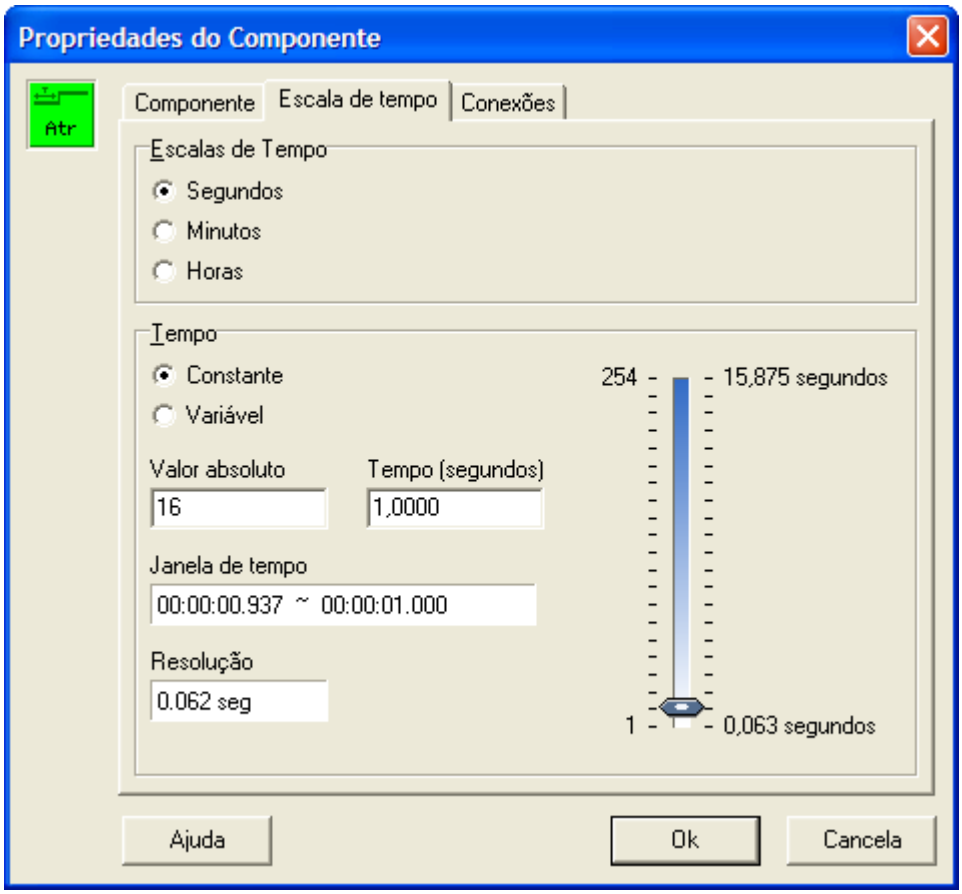
Ciclo de Execução de 1/16s (62,5ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	62,5ms	15,875s
Minutos	0,25min = 15s	63,5min = 1h 3min 30s
Horas	0,0667h = 4min	16,933h = 16h 56min

Ciclo de Execução de 1/32s (31,25ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	31,25ms	7,9375s
Minutos	0,125min = 7,5s	31,75min = 31min 45s
Horas	0,0333h = 2min	8,467h = 8h 28min

Ciclo de Execução de 1/64s (15,625ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	15,625ms	3,9688s
Minutos	0,063min = 3,75s	15,875min = 15min 52,5s
Horas	0,0167h = 1min	4,233h = 4h 14min

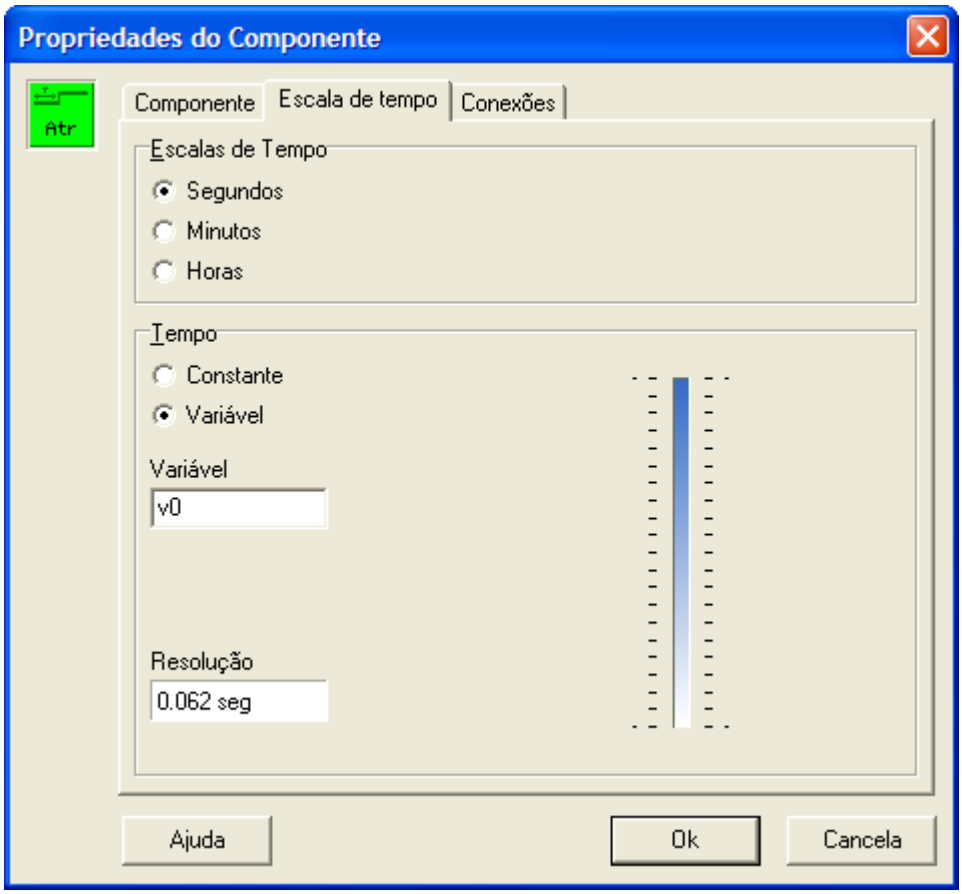
Ciclo de Execução de 1/256s (3,9063ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	3,9063ms	0,9922s
Minutos	0,016min = 0,94s	3,9688min = 3min 58,1s
Horas	0,0042h = 15s	1,058h = 1h 3min 30s

A seguir temos a tela de edição deste bloco, no caso usando uma valor constante de 1 segundo para o tempo do bloco Atraso:

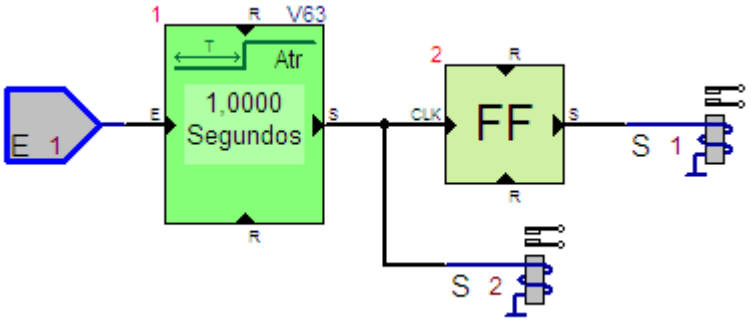


A barra de seleção do tempo pode ser arrastada com o mouse ou usar as setas do teclado do computador. Também é possível digitar o tempo desejado no quadro Tempo. O tempo digitado será arredondado para o valor mais próximo ao pressionar a tecla **[Ok]**.

Por fim, em vez de um valor constante para o tempo deste bloco, pode-se escolher uma variável que determinará o tempo do bloco Atraso:



ATENÇÃO: No caso de blocos de temporização utilizando ciclo de execução de 1/256s é necessário que o tempo de ciclo do programa aplicativo seja menor ou igual a 1/256s (3,9ms), ou o bloco de temporização sofrerá um atraso na temporização, devido ao tempo de ciclo do programa ser superior à resolução da temporização.



Exemplo de Programa Aplicativo: **Bloco_Atraso.d1g**

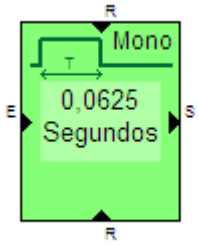
Acima temos um pequeno exemplo de uso do bloco de temporização Atraso. No caso, este bloco está sendo usado para filtrar o sinal da entrada E1 do controlador µDX100. Ele só permite que pulsos com duração superior a 1s passem para a saída S. Com isso, evita-se que sinais espúrios

venham a trocar o estado da saída S1 do µDX100 (esta saída troca de estado a cada borda de subida da saída S do bloco de Atraso graças ao flip-flop). Ou seja, é necessário que a entrada E1 fique energizada por um mínimo de 1s para ocasionar a troca de estado na saída S1. Foi incluída uma saída S2 entre o bloco de atraso e o flip-flop para observação do nodo de saída do bloco Atraso.

Importante: como todos os blocos de temporização, o bloco Atraso usa uma variável auxiliar para efetuar a contagem de tempo. Esta variável é alocada automaticamente a partir da última variável disponível no controlador (v15 no caso de µDX100, v63 no caso de µDX100+) e aparece no canto superior direito do bloco quando é efetuada a pré-compilação. Cuidado para que a lista de variáveis usadas no programa aplicativo (lista crescente) não encontre a lista de variáveis auxiliares para blocos temporizadores (lista decrescente).

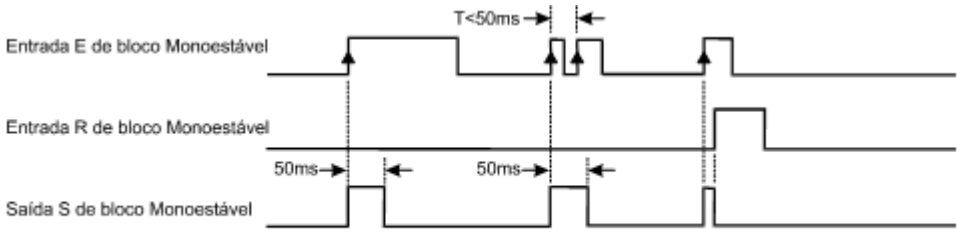
GERAL - Monoestável

Este bloco gera um pulso de largura programável no nodo de Saída S a cada borda de subida no nodo de Entrada E. Quando o nodo de entrada (E) é acionado o nodo de saída é também acionado imediatamente, mantendo-se energizado durante o tempo programado no operando Tempo. Ao contrário do bloco anterior (Atraso), o nodo de saída (S) retorna a zero após este tempo, independentemente da entrada (E) ter retornado ou não a zero. Para um novo disparo é preciso que o nodo de entrada (E) retorne a zero e volte a subir. Durante o tempo de Monoestável (saída S ativa) o bloco se torna insensível a entrada E, ou seja, se trata de Monoestável não-retrigável.



Note que, ao contrário dos blocos de Atraso, os blocos de Monoestável são sensíveis a borda de subida do nodo de Entrada E. Já os blocos de Atraso são sensíveis ao nível do nodo de Entrada E. Os nodos de Reset (R) estão internamente interligados e permitem desativar o bloco imediatamente. Note que qualquer sinal ligado a um dos nodos R está automaticamente ligado ao outro nodo R, já que se trata de dois nodos ligados internamente pelo bloco.

Abaixo temos uma representação gráfica do comportamento do bloco de Monoestável:



BLOCO MONOESTÁVEL PROGRAMADO PARA 50ms

Para especificar o tempo do bloco Monoestável pode-se utilizar uma constante ou uma variável. As faixas de duração disponíveis são: Horas, Minutos e Segundos. Os tempos variam conforme a

velocidade de execução do programa e a faixa escolhida:

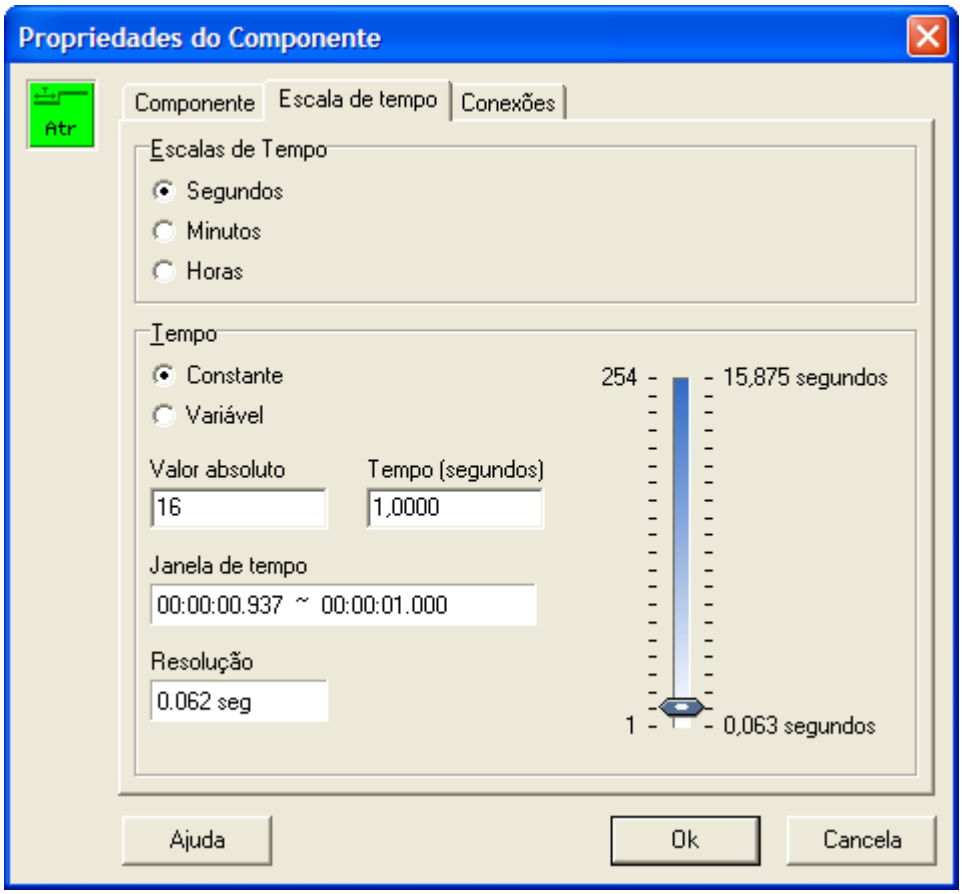
Ciclo de Execução de 1/16s (62,5ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	62,5ms	15,875s
Minutos	0,25min = 15s	63,5min = 1h 3min 30s
Horas	0,0667h = 4min	16,933h = 16h 56min

Ciclo de Execução de 1/32s (31,25ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	31,25ms	7,9375s
Minutos	0,125min = 7,5s	31,75min = 31min 45s
Horas	0,0333h = 2min	8,467h = 8h 28min

Ciclo de Execução de 1/64s (15,625ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	15,625ms	3,9688s
Minutos	0,063min = 3,75s	15,875min = 15min 52,5s
Horas	0,0167h = 1min	4,233h = 4h 14min

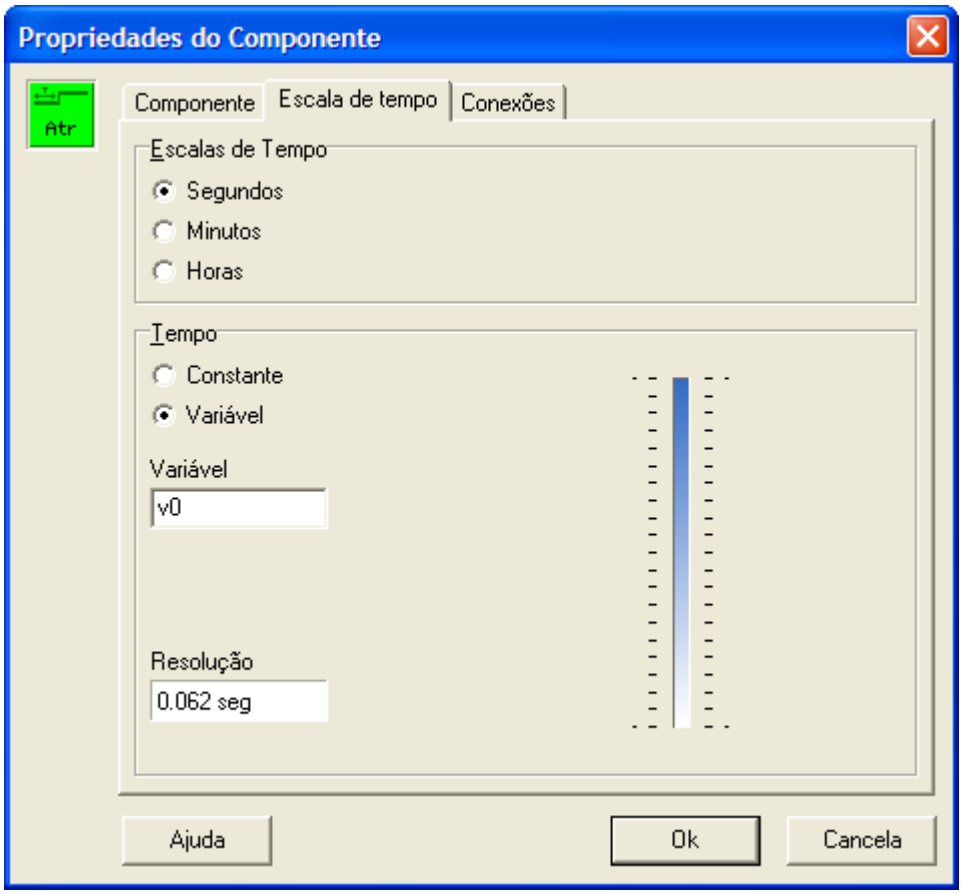
Ciclo de Execução de 1/256s (3,9063ms)		
Escala de Tempo	**Tempo Mínimo**	**Tempo Máximo**
Segundos	3,9063ms	0,9922s
Minutos	0,016min = 0,94s	3,9688min = 3min 58,1s
Horas	0,0042h = 15s	1,058h = 1h 3min 30s

A seguir temos a tela de edição deste bloco, no caso usando uma valor constante de 1 segundo para o tempo do bloco Monoestável:



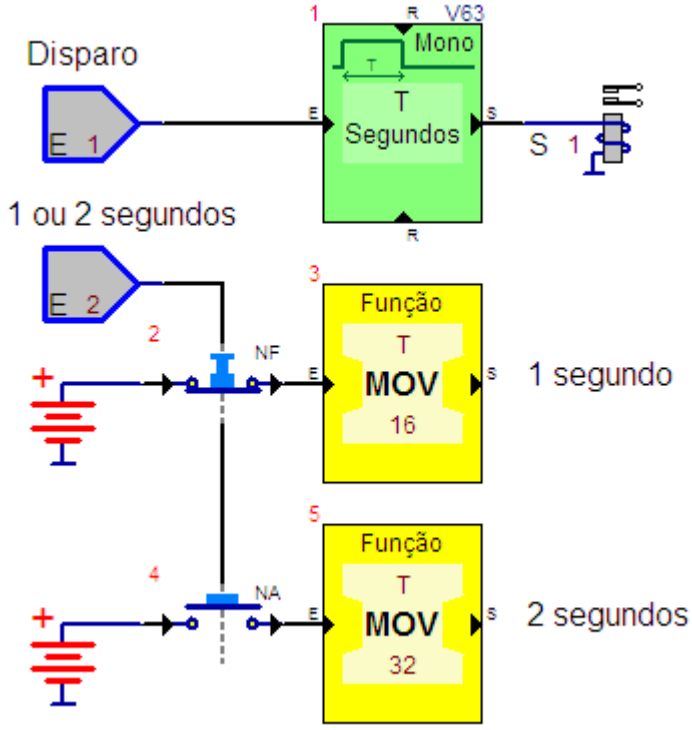
A barra de seleção do tempo pode ser arrastada com o mouse ou usar as setas do teclado do computador. Também é possível digitar o tempo desejado no quadro Tempo. O tempo digitado será arredondado para o valor mais próximo ao pressionar a tecla **[Ok]**.

Por fim, em vez de um valor constante para o tempo deste bloco, pode-se escolher uma variável que determinará o tempo do bloco Monoestável:



ATENÇÃO: No caso de blocos de temporização utilizando ciclo de execução de 1/256s é necessário que o tempo de ciclo do programa aplicativo seja menor ou igual a 1/256s (3,9ms), ou o bloco de temporização sofrerá um atraso na temporização, devido ao tempo de ciclo do programa ser superior à resolução da temporização.

A seguir temos um pequeno exemplo de uso do bloco de temporização Monoestável. No caso, este bloco está sendo usado para gerar um pulso de 1 segundo ou 2 segundos na saída S1 cada vez que a entrada E1 do µDX100 é energizada. A seleção do tempo é feita mudando-se a atribuição de valor para a variável T via entrada E2 do mesmo módulo. Caso a entrada E1 seja desacionada e novamente acionada durante o tempo do Monoestável, este pulso na entrada será ignorado, já que se trata de Monoestável não-retrigável.

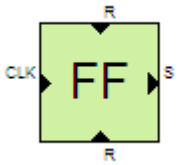


Exemplo de Programa Aplicativo: **Bloco_Mono.d1g**

Importante: como todos os blocos de temporização, o bloco Monoestável usa uma variável auxiliar para efetuar a contagem de tempo. Esta variável é alocada automaticamente a partir da última variável disponível no controlador (v15 no caso de µDX100, v63 no caso de µDX100+) e aparece no canto superior direito do bloco quando é efetuada a pré-compilação. Cuidado para que a lista de variáveis usadas no programa aplicativo (lista crescente) não encontre a lista de variáveis auxiliares para blocos temporizadores (lista decrescente).

GERAL - Flip-Flop

Este bloco implementa um flip-flop (biestável) que troca o estado do nodo de saída cada vez que o nodo de entrada é energizado. Note que qualquer sinal ligado a um dos nodos R está automaticamente ligado ao outro nodo R, já que se trata de dois nodos ligados internamente pelo bloco. Estas conexões desativam o flip-flop.



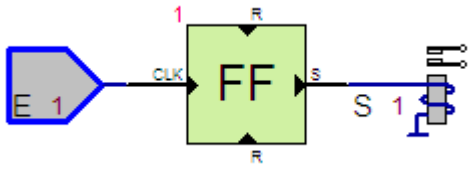
Abaixo temos uma tabela de estados para o flip-flop:

CLK	R	S
0	0	Q
1	0	Q
	0	invQ
x	1	0

Note que foi usada a seguinte simbologia:

x	Estado irrelevante
0	Estado zero
1	Estado um
	Borda de subida
invQ	Estado da saída inverte
Q	Estado da saída inalterado

Este bloco pode ser usado para gerar um comutador de estado (a cada pulso na entrada CLK troca o estado da saída). Veja o exemplo a seguir:



Exemplo de Programa Aplicativo: **Bloco_FF.d1g**

Neste exemplo a cada energização da entrada E1 do µDX100 a saída S1 do controlador inverte seu estado.

Atenção: só é possível utilizar quatro blocos Flip-Flop em um programa aplicativo, tanto no µDX100 quanto em µDX100+. No caso de usar-se um número maior deste bloco ocorrerá um erro na compilação do programa (quantidade de flip-flops excede o limite).

GERAL - Função

Este bloco de instrução é o que oferece maior quantidade de opções de configuração, permitindo escolher uma entre dez funções que incluem aritmética e lógica.

Sua operação é executada enquanto o nodo de entrada estiver ligado, sendo que o nodo de saída

será ligado sempre que a operação for completada com sucesso. Isto sempre ocorre nas operações de aritmética ou de lógica.

Todas as operações são tratadas utilizando-se dois operandos: Operando 1 e Operando 2. Nas funções aritméticas e nas lógicas o resultado do cálculo com os dois operandos é sempre retornado à variável do Operando 1. Os cálculos são sempre na seguinte ordem:

Operando 1 <função> Operando 2 → Operando 1

O Operando 1 é sempre uma variável. Já o Operando 2 pode ser uma variável ou uma constante. O resultado da operação sempre é atribuído ao Operando 1.

As operações disponíveis são as seguintes:

OP1 = OP1 + OP2 - soma
OP1 = OP1 - OP2 - subtração
OP1 = OP1 mov OP2 - move OP2 para OP1
OP1 = OP1 shr OP2 - desloca OP1 à direita por OP2 bits
OP1 = OP1 shl OP2 - desloca OP1 à esquerda por OP2 bits
OP1 = OP1 and OP2 - operação lógica AND (E)
OP1 = OP1 or OP2 - operação lógica OR (OU)
OP1 = OP1 xor OP2 - operação lógica XOR (OU EXCLUSIVO)

O µDX100 não trabalha com valores negativos, apesar dos cálculos aritméticos serem feitos em complemento de 2. Assim o resultado do cálculo 10 - 15 não será -5 e sim 251 (256 - 5 = 251). Note que todas as variáveis e constantes do µDX são de 8 bits, ou seja, assumem valores entre 0 e 255. Na edição do bloco de função é possível usar-se valores hexadecimais em vez de valores decimais para facilitar as operações que envolvam manipulação de bits. Basta colocar um caracter **h** ao final do valor. Por exemplo, 234 é equivalente a 0EAh. Quando o valor hexadecimal inicia com uma letra é preciso acrescer um zero antes para que o PG entenda tratar-se de uma constante hexadecimal e não o nome de uma variável.

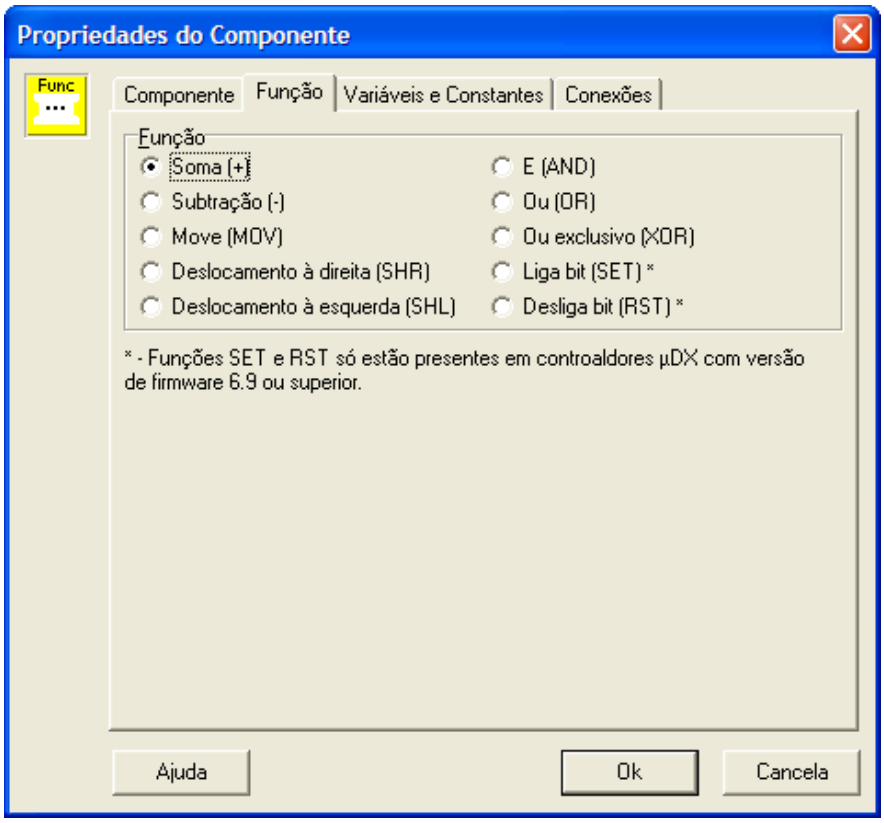
A partir da versão de firmware igual ou superior a 6.9, o Controlador µDX100 permite duas funções adicionais:

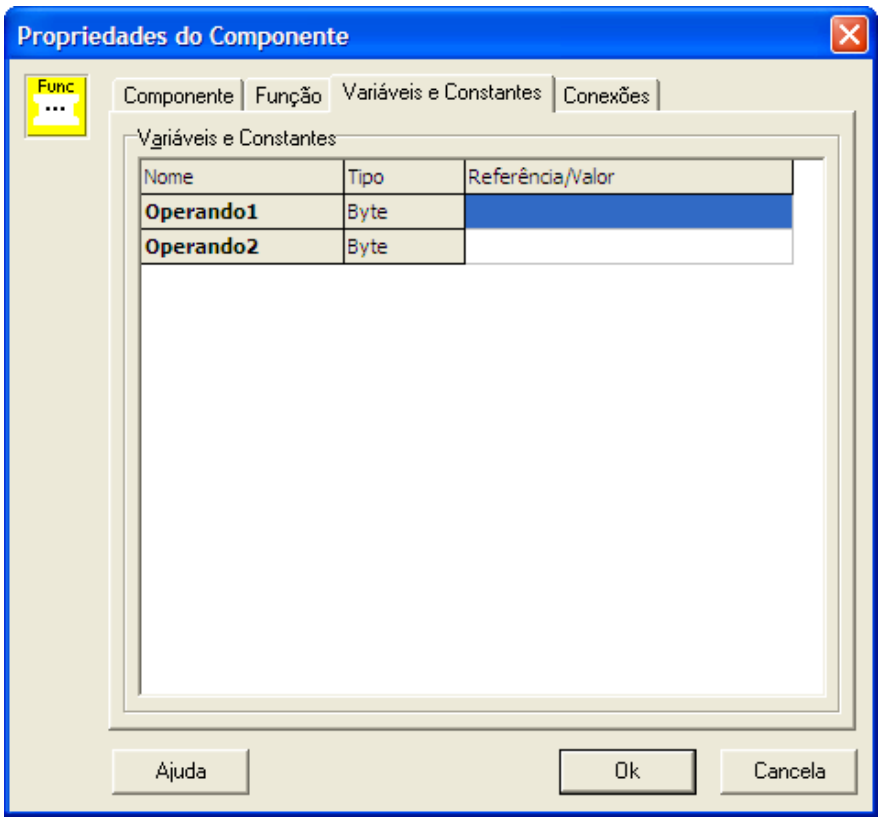
OP1 = OP1 SET OP2 - Faz o bit OP2 de OP1 = nodo de entrada.
OP1 = OP1 RST OP2 - Faz o bit OP2 de OP1 = nodo de entrada invertido.

Estas funções adicionais permitem modificar apenas um bit de determinada variável (especificada por OP1), conforme o estado do nodo de entrada do bloco. São especialmente úteis para acionar saídas da Expansão de Entradas/Saídas (lembre-se que a Expansão utiliza variáveis do µDX100 para acessar suas entradas e saídas).

Atenção: o bloco Função é executado a cada ciclo de execução do programa aplicativo, se seu nodo de entrada estiver energizado. Assim, onde for necessário executá-lo apenas uma vez a cada borda de energização do nodo de entrada é preciso intercalar um bloco de Pulso.

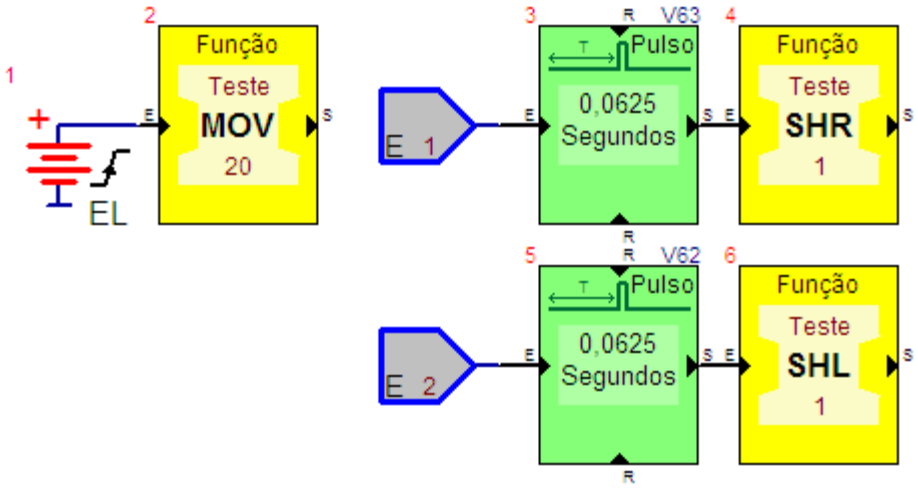
As telas de edição deste bloco permitem selecionar a operação desejada e o Operando 1 e Operando 2:





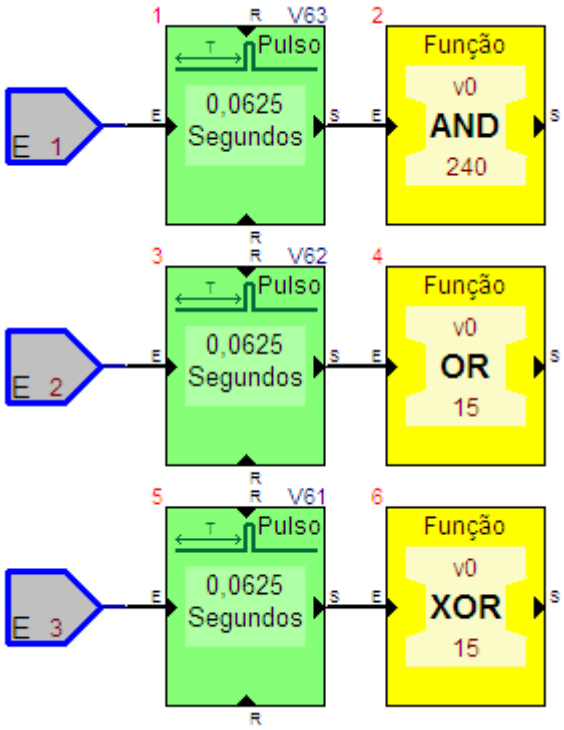
A seguir temos um pequeno exemplo de uso dos blocos de Função. A entrada E1 do controlador µDX100 efetua um operação **SHR** (shift right) de uma posição na variável Teste, enquanto a entrada E2 efetua uma operação **SHL** (shift left) de uma posição nesta variável. Ora, fazer um deslocamento de um bit para a direita equivale a dividir por dois a variável, e fazer um

deslocamento de um bit para a esquerda equivale a multiplicar por dois. Como a variável Teste foi inicializada (via bloco de Função com operação **MOV**) com valor 20, ao energizar a entrada E1 a variável Teste irá assumir valor 10. Já se for energizada a entrada E2 a variável assumirá valor 40. Veja que foram utilizados blocos de Pulso na entrada dos blocos de Função para evitar que o µDX100 execute mais de uma vez estes blocos.



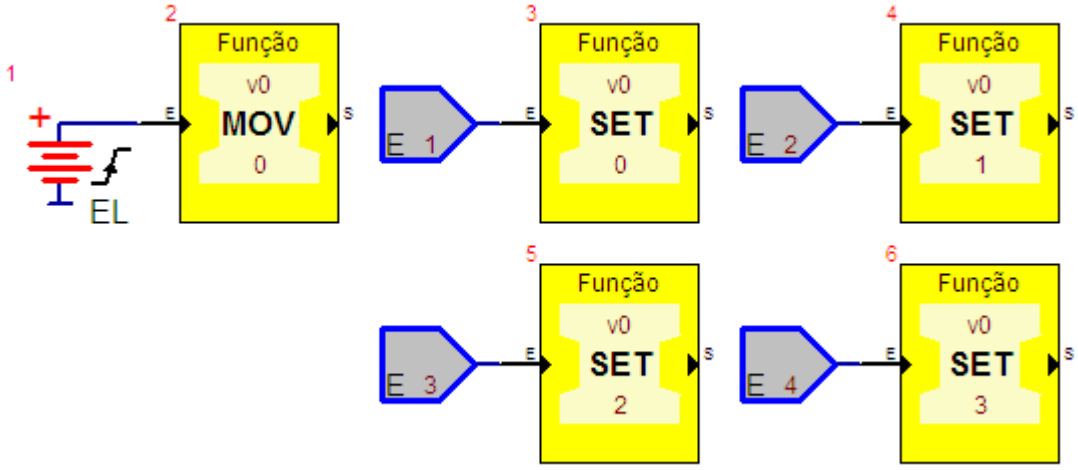
Exemplo de Programa Aplicativo: **Bloco_Funcao.d1g**

As operações **AND**, **OR** e **XOR** permitem manipular bits de variáveis do µDX100. A operação **AND** permite zerar determinados bits da variável e a operação **OR** ligar determinados bits. Já a função **XOR** permite inverter o estado de determinados bits da variável. A seguir temos um exemplo. Caso E1 seja energizada desliga-se os 4 bits inferiores da variável v0, se E2 for energizada liga-se os 4 bits inferiores, e se E3 for energizada inverte-se os 4 bits inferiores de v0.



Exemplo de Programa Aplicativo: **Bloco_Funcao2.d1g**

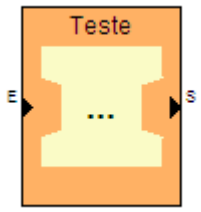
O próximo exemplo utiliza as operações SET para atribuir aos 4 bits inferiores da variável v0 o estado das quatro entradas do controlador µDX100. Ou seja, cada bit é acionado conforme o estado da entrada (E1 a E4):



Exemplo de Programa Aplicativo: **Bloco_Funcao3.d1g**

GERAL - Comparação

Este bloco de instrução oferece oito opções de configuração, entre comparação e testes.



Sua operação é executada enquanto o nodo de entrada estiver ligado, sendo que o nodo de saída será ligado sempre que o resultado é verdadeiro nas operações de comparação e de teste. Todas as comparações são tratadas utilizando-se dois operandos: Operando 1 e Operando 2. O resultado da comparação atua sobre o nodo de saída. As comparações são sempre na seguinte ordem:

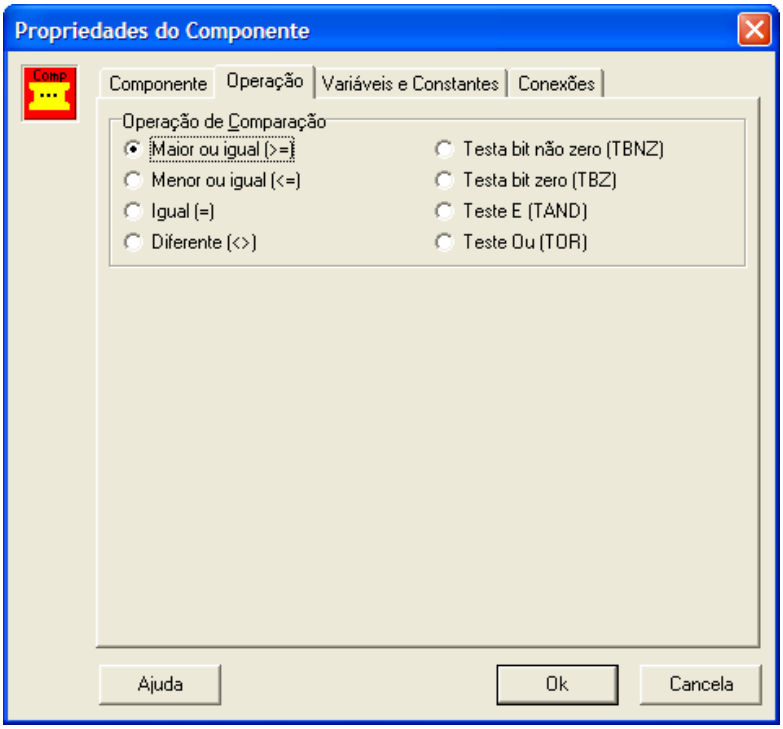
Operando 1 <função> Operando 2 → Nodo de Saída

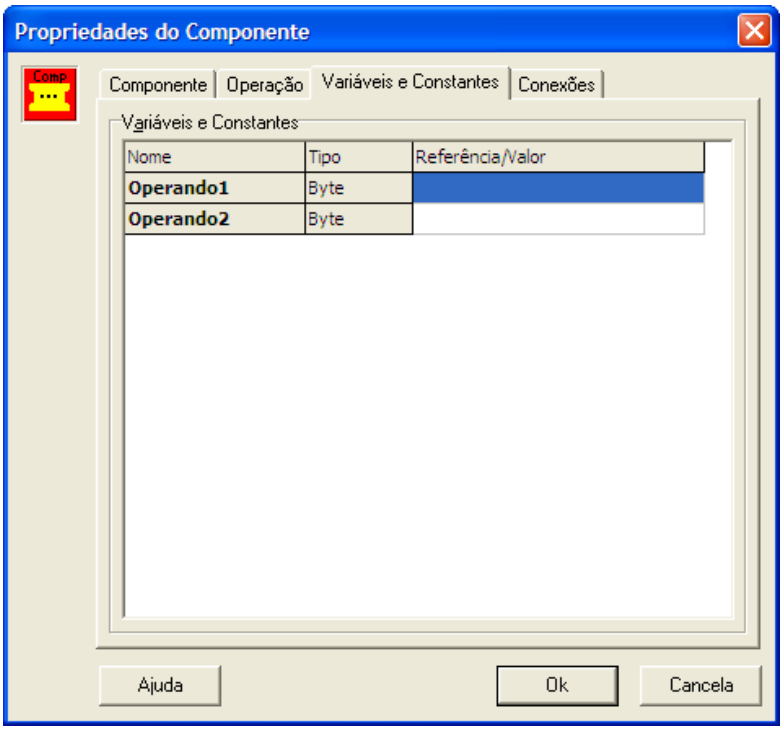
O Operando 1 é sempre uma variável. Já o Operando 2 pode ser uma variável ou uma constante. As operações disponíveis são as seguintes:

OP1 >= OP2	- liga a saída se OP1 maior ou igual a OP2
OP1 <= OP2	- liga a saída se OP1 menor ou igual a OP2
OP1 = OP2	- liga a saída se OP1 igual a OP2
OP1 <> OP2	- liga a saída se OP1 diferente de OP2
OP1 tbnz OP2	- liga a saída se o bit [OP2] de OP1 estiver em 1 (test bit not zero)
OP1 tbz OP2	- liga a saída se o bit [OP2] de OP1 estiver em 0 (test bit zero)
OP1 tand OP2	- liga a saída se todos os bits em 1 de OP2 estiverem em 1 no OP1
OP1 tor OP2	- liga a saída se pelo menos um dos bits em 1 de OP2 estiver em 1 no OP1

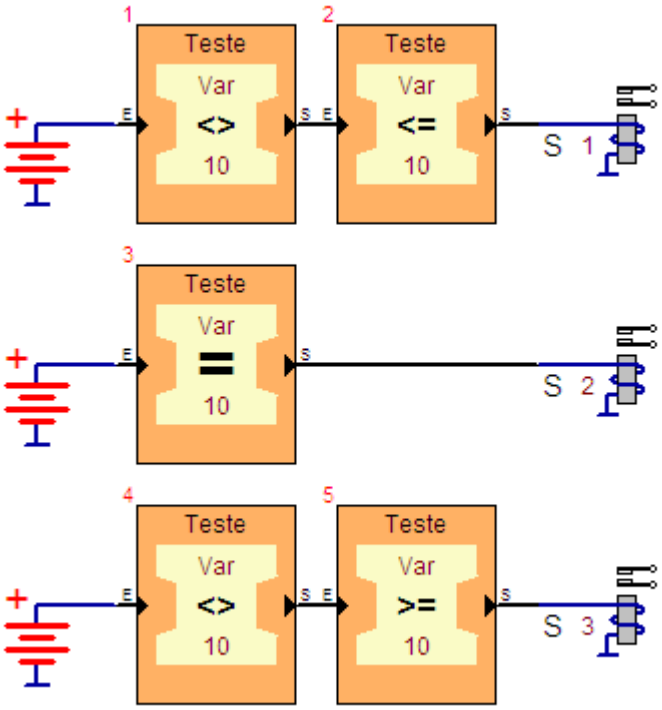
Note que todas as variáveis e constantes do µDX100 são de 8 bits, ou seja, assumem valores entre 0 e 255. Na edição do bloco de comparação é possível usar-se valores hexadecimais em vez de valores decimais para facilitar as operações que envolvam manipulação de bits. Basta colocar um caracter **h** ao final do valor. Por exemplo, 234 é equivalente a 0EAh. Quando o valor hexadecimal inicia com uma letra é preciso acrescer um zero antes para que o PG entenda tratar-se de uma constante hexadecimal e não o nome de uma variável.

As telas de edição deste bloco permitem selecionar a operação desejada e o Operando 1 e Operando 2:



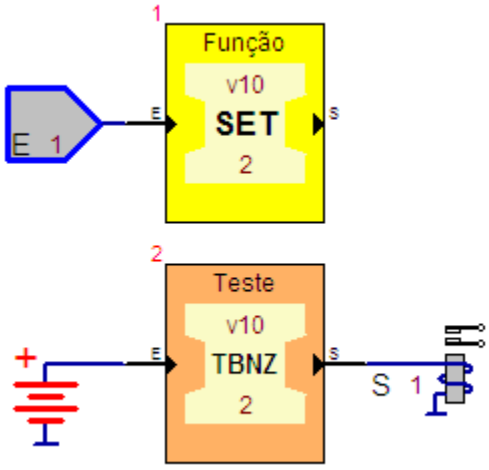


O exemplo a seguir demonstra o uso dos blocos de comparação para verificação se a variável Var é maior, menor, ou igual a 10. Como não existe operação de menor (**<**) ou maior (**>**) se utiliza o bloco de diferente(**<>**) com os blocos de maior ou igual (≥) e menor ou igual (≤) para restringir o teste:



Exemplo de Programa Aplicativo: **Bloco_Comparacao.d1g**

O exemplo seguinte foi elaborado apenas para demonstrar o teste de bit. A entrada E1 do controlador µDX100 é associada ao bit 2 da variável v10. Este bit é testado pelo bloco de comparação **TBNZ** (test bit not zero) e, no caso do bit estar ligado, é acionada a saída S1 do controlador.



Exemplo de Programa Aplicativo: **Bloco_Comparacao2.d1g**

GERAL - Função Word

Este bloco de instrução, operacional apenas em controladores µDX101 (versão 9.6 ou superior), permite operações aritméticas e lógica em 16 bits sem sinal (word).

Sua operação é executada enquanto o nodo de entrada estiver ligado, sendo que o nodo de saída será ligado sempre que a operação for completada com sucesso. Isto sempre ocorre nas operações de aritmética ou de lógica.

Todas as operações são tratadas utilizando-se dois operandos: Operando 1 e Operando 2. Nas funções aritméticas e nas lógicas o resultado do cálculo com os dois operandos é sempre retornado à variável do Operando 1. Os cálculos são sempre na seguinte ordem:

Operando 1 <função> Operando 2 → Operando 1

O Operando 1 é sempre uma variável. Já o Operando 2 pode ser uma variável ou uma constante. O resultado da operação sempre é atribuído ao Operando 1. No bloco de Função Word sempre é alocada a variável apontada pelo bloco e a subseqüente, de forma a formar o word de 16 bits (note que as variáveis no µDX101 são sempre de 8 bits). Assim, se indicarmos a variável v10, por exemplo, em uma função word, será utilizada a variável v10 (byte LSB do word) e a variável v11 (byte MSB do word).

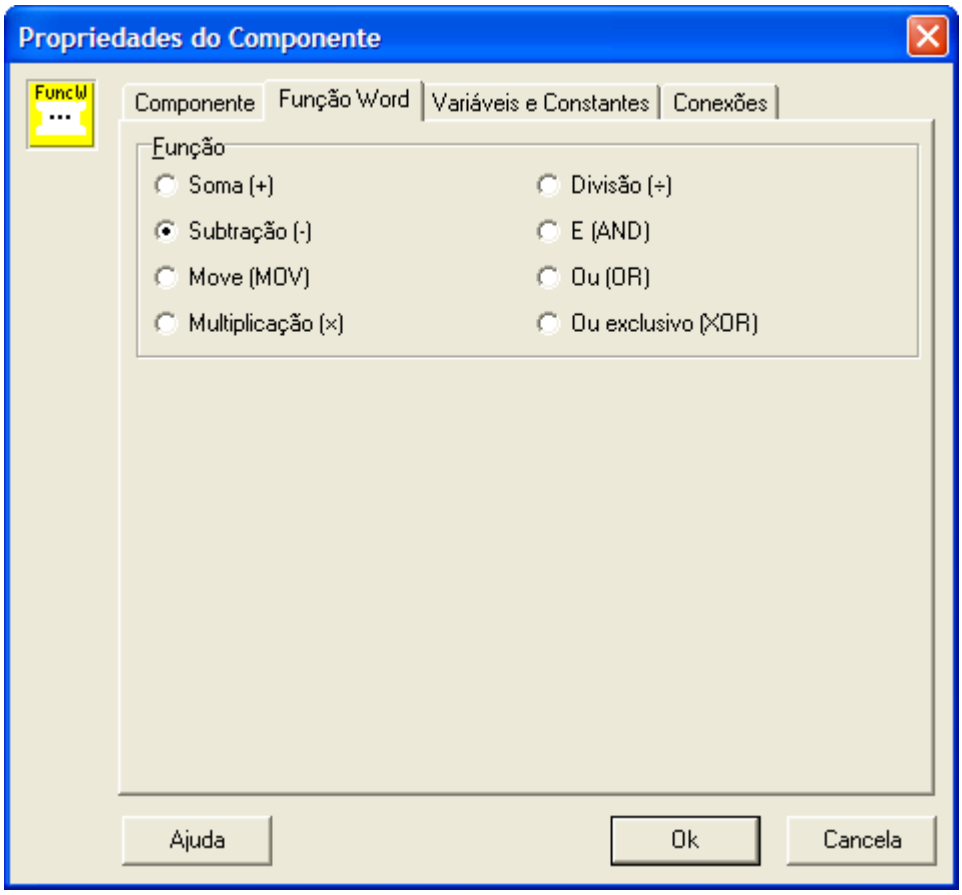
As operações disponíveis são as seguintes:

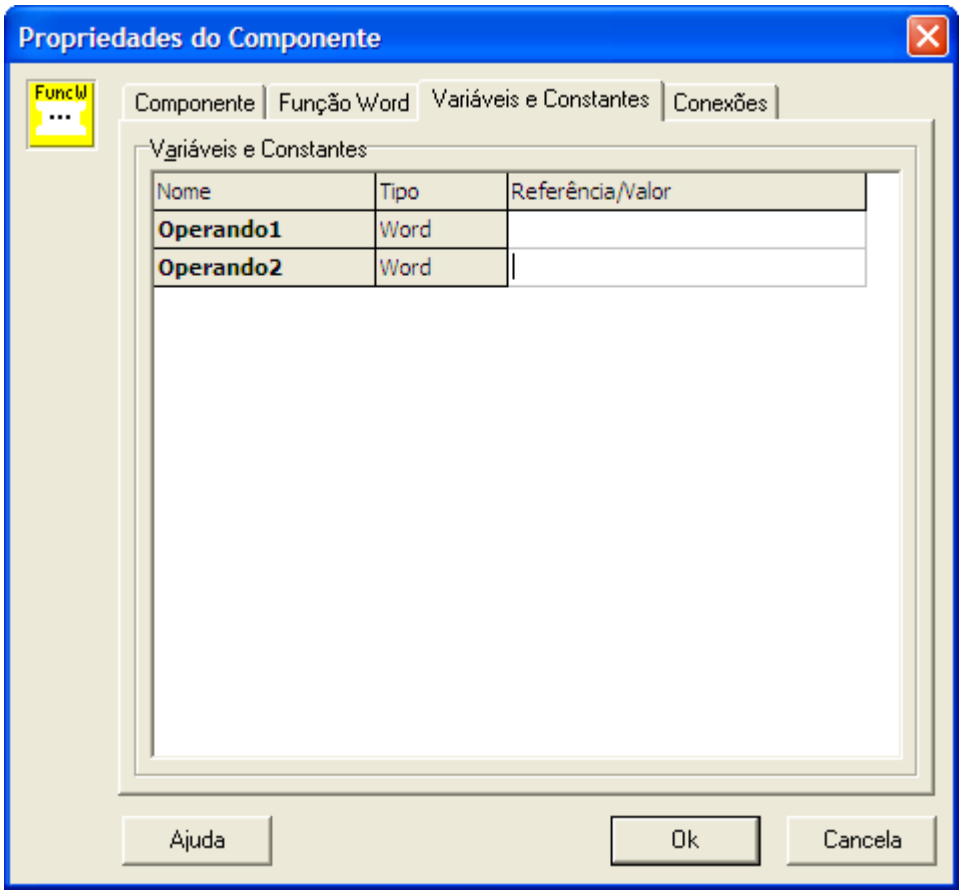
OP1 = OP1 + OP2	- soma
OP1 = OP1 - OP2	- subtração
OP1 = OP1 mov OP2	- move OP2 para OP1
OP1 = OP1 × OP2	- multiplicação
OP1 = OP1 ÷ OP2	- divisão
OP1 = OP1 and OP2	- operação lógica AND (E)
OP1 = OP1 or OP2	- operação lógica OR (OU)
OP1 = OP1 xor OP2	- operação lógica XOR (OU EXCLUSIVO)

O µDX101 não trabalha com valores negativos, apesar dos cálculos aritméticos serem feitos em complemento de 2. Assim o resultado do cálculo 10 - 15 não será -5 e sim 65531 (65536 - 5 = 65531). Note que todas as variáveis e constantes do µDX101 em blocos word são de 16 bits, ou seja, assumem valores entre 0 e 65535. Na edição do bloco de função é possível usar-se valores hexadecimais em vez de valores decimais para facilitar as operações que envolvam manipulação de bits. Basta colocar um caracter **h** ao final do valor. Por exemplo, 234 é equivalente a 0EAh. Quando o valor hexadecimal inicia com uma letra é preciso acrescer um zero antes para que o PG entenda tratar-se de uma constante hexadecimal e não o nome de uma variável.

Atenção: o bloco Função é executado a cada ciclo de execução do programa aplicativo, se seu nodo de entrada estiver energizado. Assim, onde for necessário executá-lo apenas uma vez a cada borda de energização do nodo de entrada é preciso intercalar um bloco de Pulso.

As telas de edição deste bloco permitem selecionar a operação desejada e o Operando 1 e Operando 2:

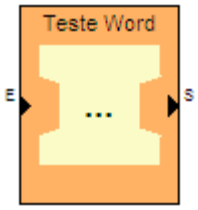

Propriedades do Componente
FuncW
Componente
Função Word
Variáveis e Constantes
Conexões
Função
Soma (+)
Subtração (-)
Move (MOV)
Multiplicação (×)
Divisão (÷)
E (AND)
Ou (OR)
Ou exclusivo (XOR)
Ajuda
Ok
Cancela



Atenção: A partir da versão 9.6 do controlador µDX101 já é permitida operação direta em word (16 bits sem sinal) usando os blocos Função Word e Comparação Word. Controladores µDX101 de versão anteriores ou controladores µDX100 não permitem manipulação direta em 16 bits. Neste caso consulte a seção Operações Aritméticas com mais de 8 bits.

GERAL - Comparação Word

Este bloco de instrução oferece oito opções de configuração, entre comparação e testes.



Sua operação é executada enquanto o nodo de entrada estiver ligado, sendo que o nodo de saída será ligado sempre que o resultado é verdadeiro nas operações de comparação e de teste. Todas as comparações são tratadas utilizando-se dois operandos: Operando 1 e Operando 2. O resultado da comparação atua sobre o nodo de saída. As comparações são sempre na seguinte ordem:

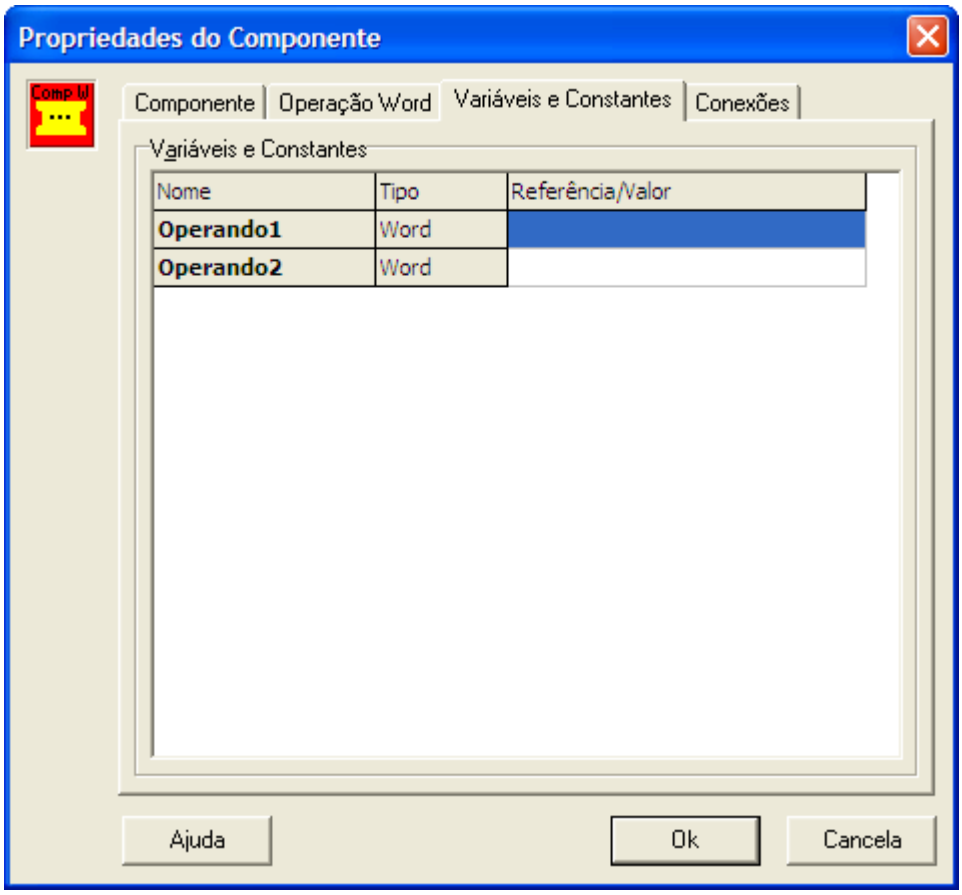
Operando 1 <função> Operando 2 → Nodo de Saída

O Operando 1 é sempre uma variável. Já o Operando 2 pode ser uma variável ou uma constante. As operações disponíveis são as seguintes:

OP1 >= OP2 - liga a saída se OP1 maior ou igual a OP2
OP1 <= OP2 - liga a saída se OP1 menor ou igual a OP2
OP1 = OP2 - liga a saída se OP1 igual a OP2
OP1 <> OP2 - liga a saída se OP1 diferente de OP2
OP1 tbnz OP2 - liga a saída se o bit [OP2] de OP1 estiver em 1 (test bit not zero)
OP1 tbz OP2 - liga a saída se o bit [OP2] de OP1 estiver em 0 (test bit zero)
OP1 > OP2 - liga a saída se OP1 maior a OP2
OP1 < OP2 - liga a saída se OP1 menor a OP2

No bloco de Comparação Word sempre é alocada a variável apontada pelo bloco e a subseqüente, de forma a formar o word de 16 bits (note que as variáveis no µDX101 são sempre de 8 bits). Assim, se indicarmos a variável v10, por exemplo, em uma comparação word, será utilizada a variável v10 (byte LSB do word) e a variável v11 (byte MSB do word).. Na edição do bloco de comparação é possível usar-se valores hexadecimais em vez de valores decimais para facilitar as operações que envolvam manipulação de bits. Basta colocar um caracter **h** ao final do valor. Por exemplo, 234 é equivalente a 0EAh. Quando o valor hexadecimal inicia com uma letra é preciso acrescer um zero antes para que o PG entenda tratar-se de uma constante hexadecimal e não o nome de uma variável.

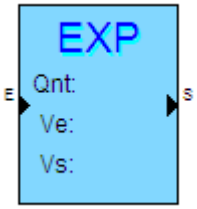
As telas de edição deste bloco permitem selecionar a operação desejada e o Operando 1 e Operando 2:



Atenção: A partir da versão 9.6 do controlador µDX101 já é permitida operação direta em word (16 bits sem sinal) usando os blocos Função Word e Comparação Word. Controladores µDX101 de versão anteriores ou controladores µDX100 não permitem manipulação direta em 16 bits. Neste caso consulte a seção Operações Aritméticas com mais de 8 bits.

GERAL - Expansão

O bloco de expansão serve para que se possa associar variáveis as entradas e saídas de uma Expansão de Entradas/Saídas para µDX100. Sem esta associação não é possível ler as entradas ou acionar as saídas da Expansão. Como a Expansão possui 8 entradas e 8 saídas cada bit da variável associada representa uma entrada ou saída.



Quando o nodo de entrada é ligado o conteúdo da variável de saída é enviado à expansão ao mesmo tempo que dela é lido um valor de 8 bits para dentro da variável de entrada.

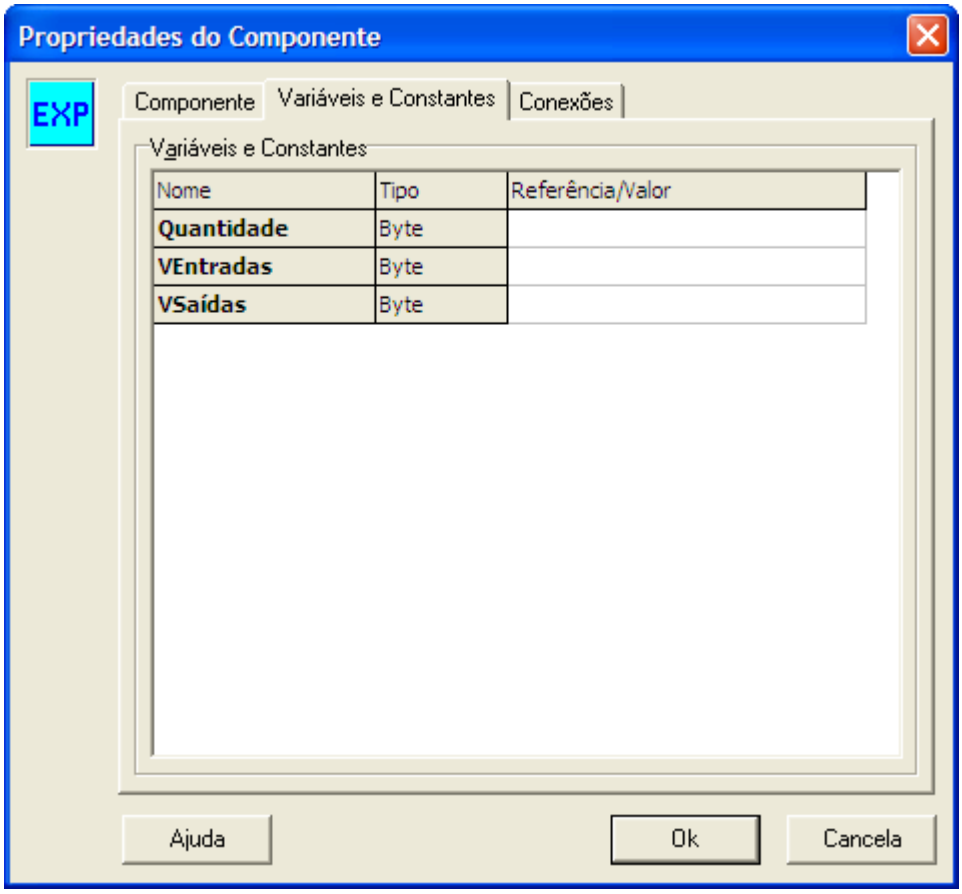
Se o mesmo número de variável for utilizado para a entrada e para a saída na expansão, no momento em que o nodo de entrada é ligado o valor atual contido na variável é transferido para a

saída da expansão e, ao término da operação, esta variável receberá os 8 bits da entrada da expansão.

No caso de controladores µDX100+ é possível especificar até 4 Expansões conectadas ao controlador. Neste caso se especifica a variável de entrada e o PG aloca as variáveis subseqüentes para as demais entradas. Da mesma forma para as saídas. Por exemplo, se for declarado neste bloco duas Expansões, e a variável de entrada for v0 e a variável de saída v10, então serão usadas as variáveis v0 e V1 para as entradas das Expansões, e v10 e v11 para as saídas das Expansões. Se for declarado o uso de 4 Expansões, então teremos v0, v1, v2 e v3 para as entradas, e v10, v11, v12 e v13 para as saídas.

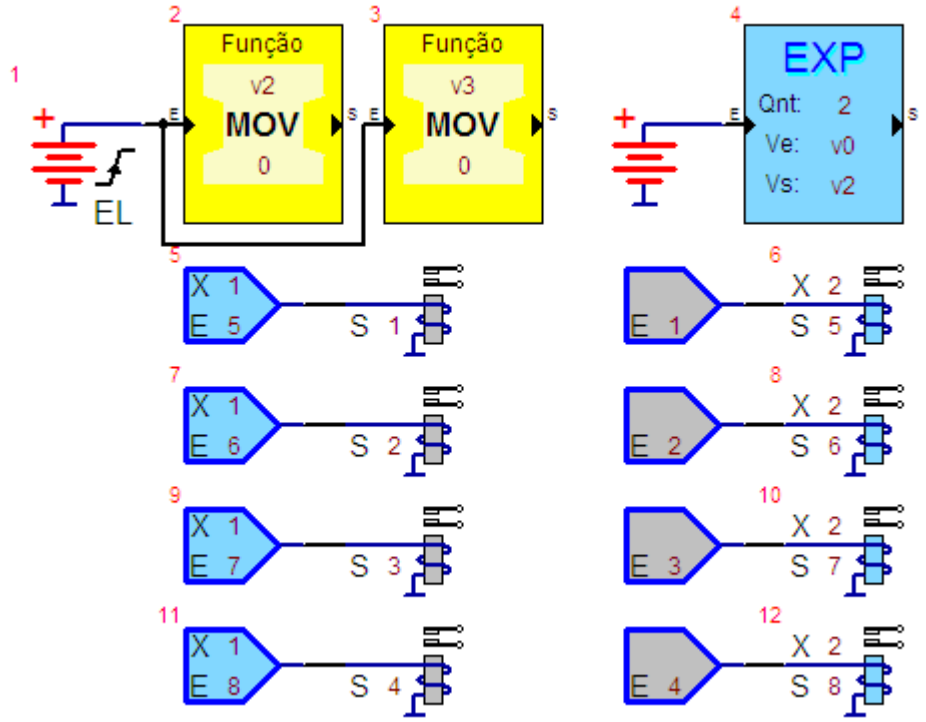
Normalmente, se usa apenas um bloco **EXP** por programa aplicativo, uma vez que ele apenas declara quantas Expansões estão sendo usadas e quais variáveis serão alocadas para o funcionamento das Expansões. As entradas e saídas das Expansões podem ser utilizadas normalmente no programa aplicativo do µDX100 usando os blocos **Entrada da Expansão** e **Saída da Expansão**. As entradas da primeira Expansão são **E5** a **E12** e as saídas **S5** a **S12** da Expansão 1 (**X1**); da segunda Expansão são **E5** a **E12** e as saídas **S5** a **S12** da Expansão 2 (**X2**); e assim em diante.

A tela de edição deste bloco permite selecionar quantidade de expansões, a variável inicial para entradas e variável inicial para saídas:



ATENÇÃO: A fonte de alimentação que acompanha o controlador µDX100 possui capacidade de corrente para suprir até 12 relés acionados simultaneamente (4 do µDX e 8 de uma Expansão). No caso de uso de múltiplas Expansões é necessário verificar se o programa aplicativo não excede esta capacidade em algum momento ou substituir a fonte de alimentação por uma mais potente (para acionar todos os 36 relés do µDX100+ e das 4 Expansões a fonte deve suprir 12V @ 1,5A (18 W).

ATENÇÃO: cuidado para não usar as variáveis utilizadas para as Expansões para outras finalidades no programa aplicativo. Modificações em variáveis associadas à entradas das Expansões serão sobre-escritas com o valor lido nestas entradas, e modificações em variáveis associadas à saídas das Expansões irão acionar os relés destes módulos. Também é preciso cuidado para deixar o devido espaço entre variáveis de entrada e de saída, no caso de múltiplas Expansões, de forma que não haja sobreposição nas listas. Por exemplo, com quatro Expansões e variável de entrada v0 a variável de saída mínima seria v4 (já que vo, v1, v2 e v3 estão alocadas para as entradas). Por fim, como a alocação de variáveis para as Expansões é seqüencial e automática deve-se especificar variáveis absolutas (Vn) para as entradas e saídas das Expansões.



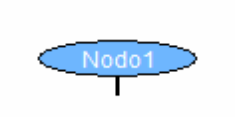
Exemplo de Programa Aplicativo: **Bloco_EXP.d1g**

O programa acima utiliza duas Expansões, sendo que as entradas E5 a E8 da primeira Expansão comandam as saídas S1 a S4 do controlador µDX100. Já as entradas E1 a E4 do µDX100 comandam as saídas S5 a S8 da segunda Expansão. Note que sempre convém inicializar as variáveis de saída das Expansões, uma vez que as variáveis no µDX100 iniciam com valor 255 (o que resultaria em todas as saídas ligadas). Além disso, os blocos de inicialização devem ser posicionados de forma que tenham número de bloco menor que o do bloco EXP. Isso irá garantir que a inicialização seja executada antes de escrever nas Expansões, evitando acionamento espúrio das saídas ao rodar o programa aplicativo.

ATENÇÃO: No caso de uso de Expansões µDX110 é possível agregar até 8 expansões, sendo que cada Expansão possui 4 entradas e 4 saídas. Neste caso deve ser especificadas as entradas e saídas de 1 a 4, e não de 5 a 12, como no caso da Expansão do µDX100 (recurso disponível a partir da versão 2.0.2.0 do PG).

GERAL - Nodo

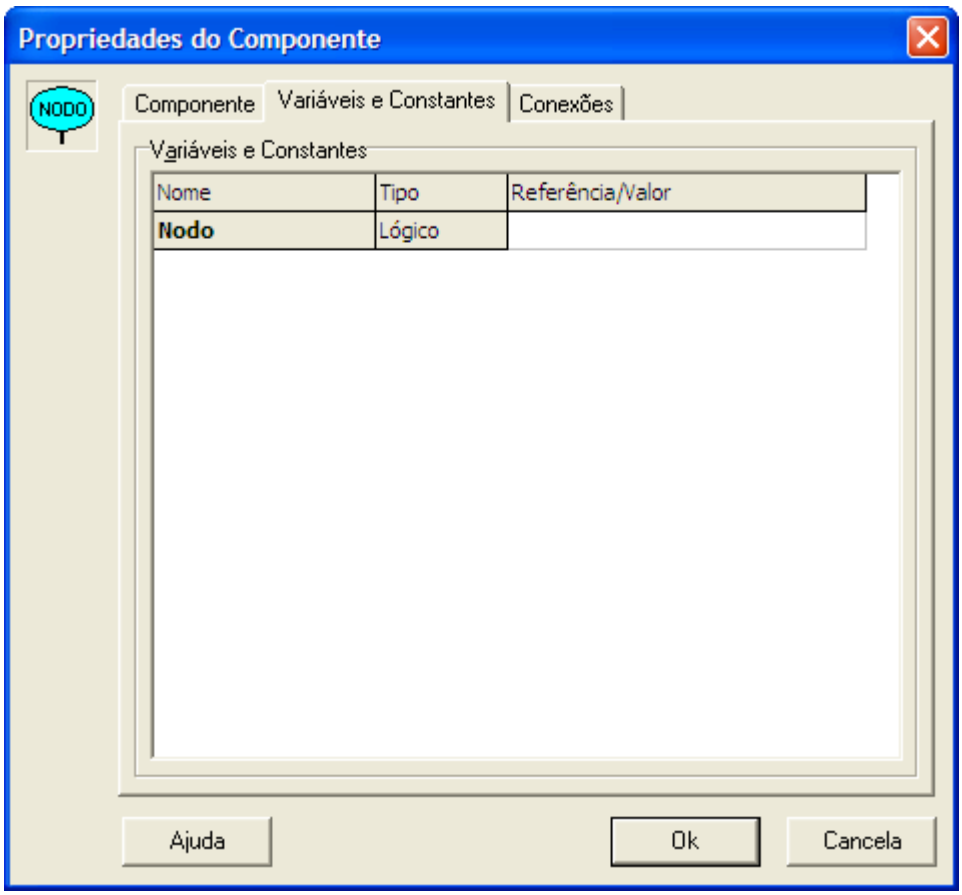
Este bloco permite nomear um nodo, ou seja, uma variável binária (apenas dois estados, ligado ou desligado), ou atribuir o valor de uma constante a este nodo (1 para ligado ou 0 para desligado), ou ainda especificar um número de nodo específico para uma conexão binária.



Com isso, é possível conectar pontos distantes do programa, ou vários programas escritos em diferentes páginas do projeto. Além disso, usando-se a nomenclatura Nxxx, sendo xxx o número do nodo, é possível especificar um número de nodo determinado para a conexão. Isso é importante se esta conexão deve ser acessível para outros controladores µDX100 em rede DXNET.

Lembre-se que os nodos de 0 a 7 (N0 a N7) são nodos de entradas e saídas no µDX100 e, portanto, possuem aplicações específicas. Também o nodo 62 (sempre desligado: GND) e o nodo 63 (sempre ligado: +V) têm função própria. Assim, não podem ser usados como nodos de conexão no programa aplicativo.

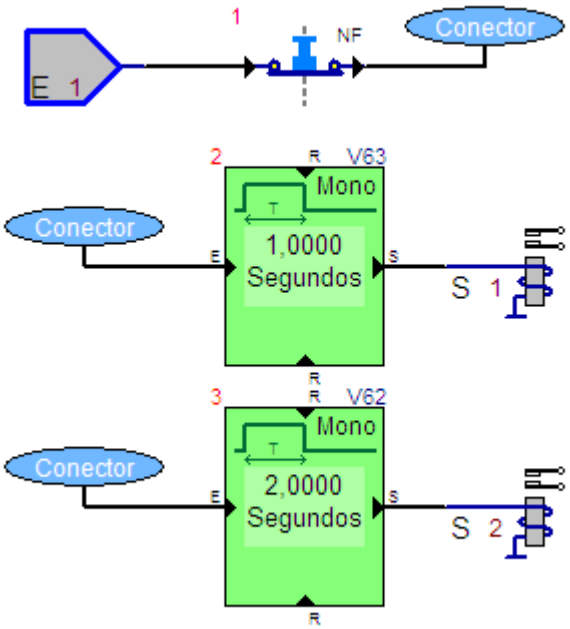
A tela de edição deste bloco permite indicar um nome para o nodo, ou especificar um nodo absoluto (Nxxx):



Note que no menu principal do Editor PG existe uma entidade chamada Rótulo, cuja função também é conectar diferentes pontos do programa entre si.

Entretanto, o rótulo apenas conecta entre si todos os pontos com o mesmo número de rótulo (de 0 a 999), não atribuindo nenhum valor determinado ou nome para esta conexão.

O exemplo a seguir mostra possível aplicação para este bloco. O programa conecta todos os nodos chamados de "Conector" entre si. Note que ao acionar a entrada E1 do µDX100 as saídas S1 e S2 são acionadas, a primeira por 1 segundo e a segunda por 2 segundos.



Exemplo de Programa Aplicativo: **Bloco_Nodo.d1g**

GERAL - Nodo ED (Energia Desliga)

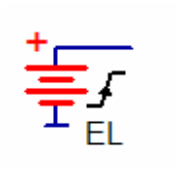
Este bloco produz um único pulso no nodo de saída, com duração de um ciclo de execução do programa aplicativo, quando o controlador programável µDX100 é desenergizado.

O Nodo ED produz o pulso no momento em que o suprimento de energia (fonte de alimentação) for desligado do µDX100. Naturalmente é preciso que o µDX esteja recebendo energia das pilhas para que continue funcionando e reconheça estas condições. A principal aplicação é permitir que o programa desenhado tenha como saber quando faltou a energia elétrica, já que as saídas (relés) não funcionam sem esta energia. Neste caso o programa pode assumir um outro estado operacional ou memorizar a ocorrência para processamento posterior. A duração do pulso é de apenas um ciclo do µDX100.

Note-se que sendo um bloco de configuração o Nodo ED não ocupa espaço de memória de programa do µDX100.

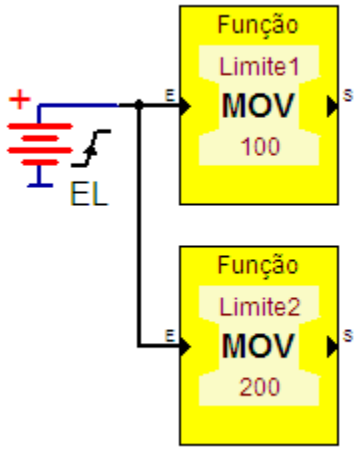
GERAL - Nodo EL (Energia Liga)

Este bloco produz um único pulso no nodo de saída, com duração de um ciclo de execução do programa aplicativo, quando o controlador programável µDX100 é energizado.



O Nodo EL produz o pulso no momento em que o suprimento de energia (fonte de alimentação) for ligado ao µDX100.

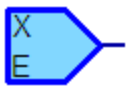
O bloco Nodo EL pode ser utilizado para inicializar variáveis no programa do µDX100. Note que o µDX100 inicia todas as variáveis com valor 255, o que muitas vezes não é conveniente. Neste caso podemos usar este bloco para gerar um pulso quando o CLP for energizado, e este pulso acionar blocos de atribuição de variáveis. Veja o exemplo abaixo, que inicializa as variáveis Limite1 e Limite2 com os valores 100 e 200, respectivamente. Note que o bloco EL inicializa as variáveis sempre que o controlador é energizado.



Exemplo de Programa Aplicativo: **Bloco_EL.d1g**

GERAL - Entrada de Expansão

Este bloco acessa as entradas digitais das Expansões de Entradas/Saídas do controlador programável µDX100. Assim como indicado na tampa superior da Expansão, cada entrada tem uma numeração própria: E5 a E12. O bloco permite edição para especificar qual entrada está sendo usada, e qual das 4 Expansões se refere (apenas para µDX100+). A conexão de saída deste bloco, como de todos os blocos do µDX100, é um nodo, ou seja, uma variável binária (apenas dois estados, ligado ou desligado).

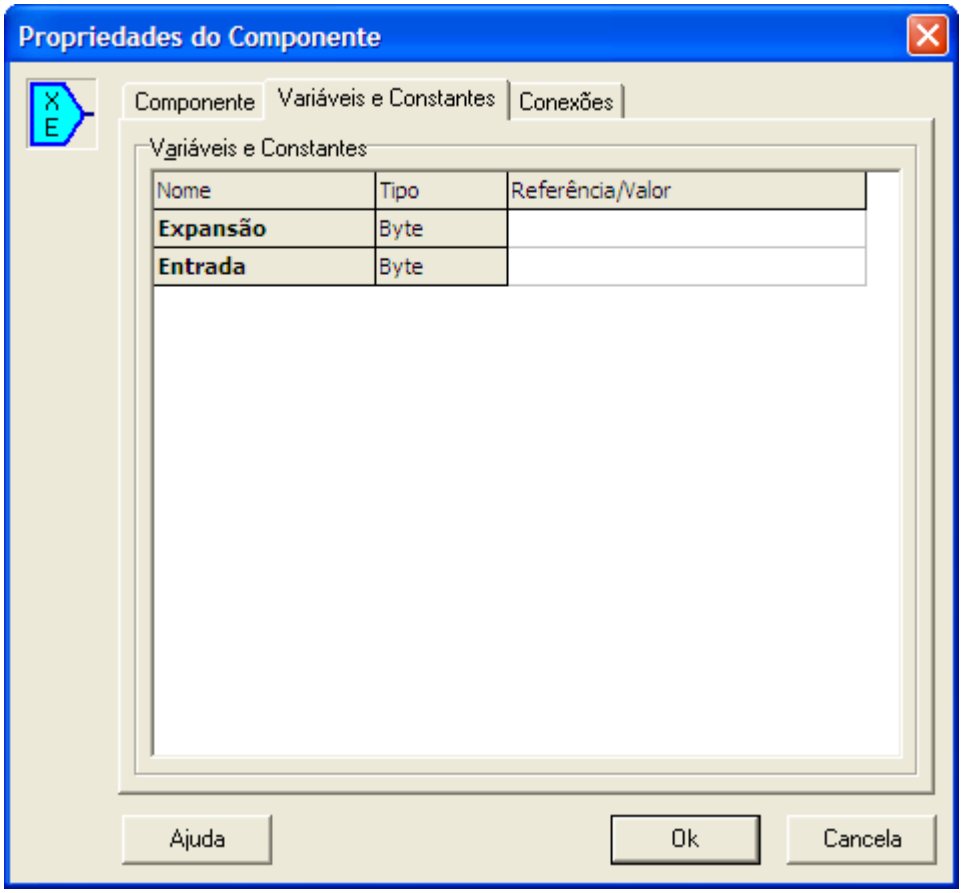


Este bloco efetivamente é contabilizado como um bloco de programação, ao contrário das entradas do controlador µDX100, que acessam nodos específicos e, por isso, não são contadas como blocos. É preciso usar um bloco **Expansão** para alocar as variáveis associadas as entradas

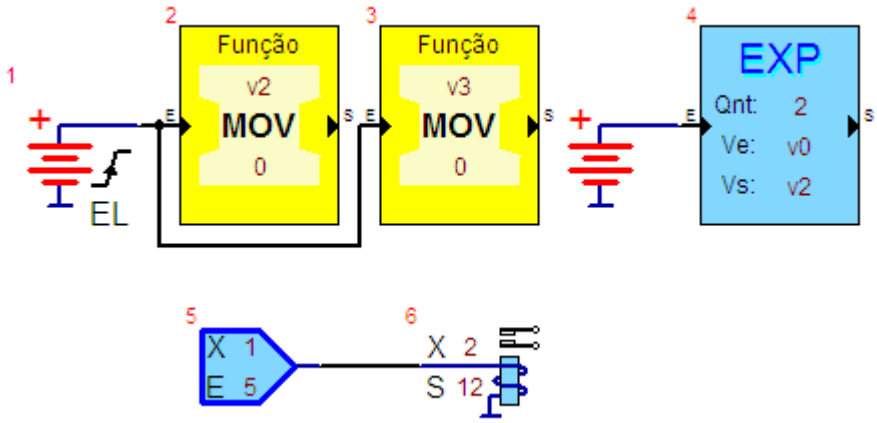
e saídas das Expansões. Sem o bloco de **Expansão** no programa aplicativo este bloco de **Entrada de Expansão** é inoperante.

ATENÇÃO: As entradas da Expansão de Entradas/Saídas para µDX100 são isoladas galvanicamente. Portanto, é possível conectar os fios da energia elétrica domiciliar diretamente em qualquer das entradas da Expansão. Apenas se certifique que os jumpers internos da Expansão estão para alta tensão (vide Expansão de Entradas/Saídas e também Expansão µDX110).

Ao editar o bloco (aponte com o mouse para o bloco e pressione a tecla direita do mouse, ou pressione a tecla de espaço no teclado do computador) surge uma tela para inserção da entrada e expansão a ser usada. Para o número da Expansão digite um valor entre 1 e 4 (desde que seja µDX100+, µDX100 permite apenas uma expansão), e para a entrada um valor de 5 a 12:



O exemplo a seguir liga a saída S12 da Expansão 2 quando a entrada E5 da Expansão 1 for energizada. Note que no caso de entradas e saídas das Expansões podem ser ligadas diretamente, sem intercalar chaves NF para isolar os blocos (como no caso de entradas e saídas do controlador µDX100). Foi usado um bloco de Nodo EL para inicializar as variáveis de saída das Expansões (v2 e v3) em zero, e estes blocos de inicialização (blocos 2 e 3) são anteriores ao bloco Expansão (bloco 4). Com isso se evita um acionamento momentâneo das saídas das Expansões ao reiniciar o CLP.



Exemplo de Programa Aplicativo: **Bloco_entrada_EXP.d1g**

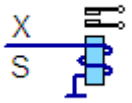
ATENÇÃO: No caso de uso de Expansões µDX110 é possível agregar até 8 expansões, sendo que cada Expansão possui 4 entradas e 4 saídas. Neste caso deve ser especificadas as entradas e saídas de 1 a 4, e não de 5 a 12, como no caso da Expansão do µDX100 (recurso disponível a partir da versão 2.0.2.0 do PG).

Veja também:
GERAL - Saída de Expansão

GERAL - Saída de Expansão

As oito saídas da Expansão para µDX100 - S5 a S12 - são feitas utilizando-se relés, cada um com um contato reversor, isto é, existe neles uma lâmina móvel que permite passar a corrente elétrica quando está encostando em um dos dois contatos elétricos montados de cada lado. Assim, quando o relé está desligado a lâmina fica encostando em um deles (Normal Fechado - NF) e afastada do outro (Normal Aberto - NA). Quando o relé é ligado a conexão se reverte, ficando o primeiro contato em aberto e o segundo fechado.

Como os dois contatos e a lâmina central estão disponíveis para conexões externas (através do conector lateral da Expansão) ambos podem ser empregados para que se tenha corrente elétrica quando o relé está desligado (usando o contato NF) ou quando está ligado (usando o contato NA).

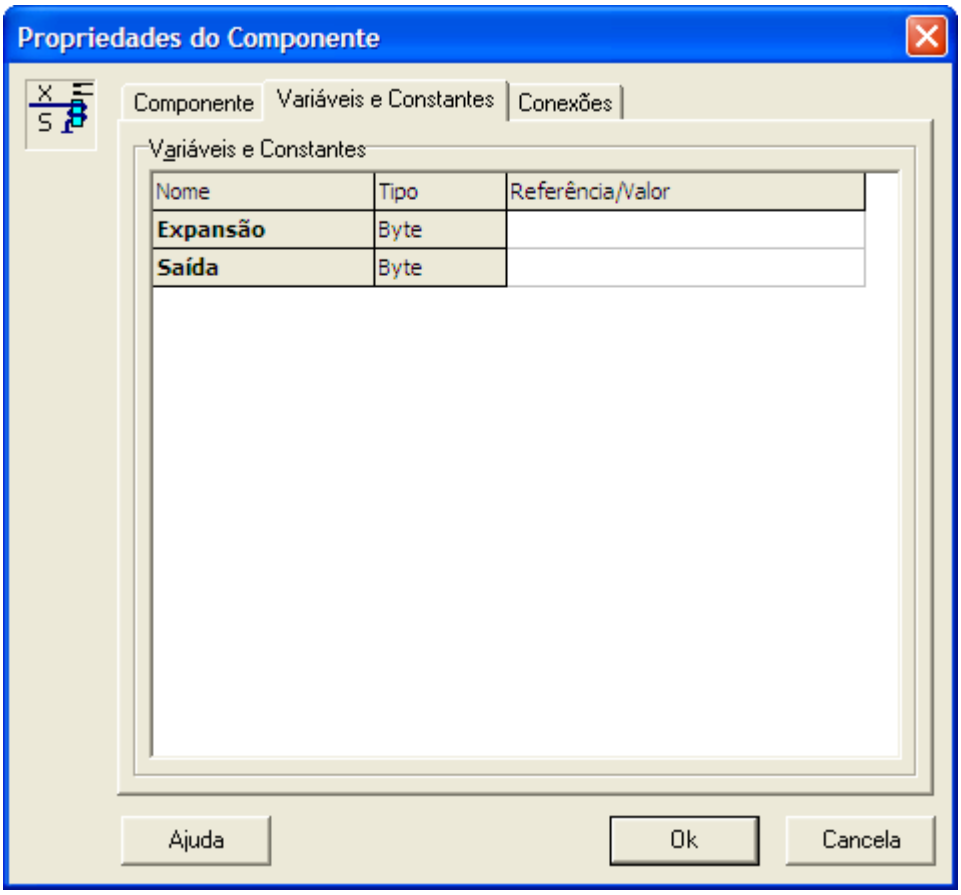


Este bloco efetivamente é contabilizado como um bloco de programação, ao contrário das saídas do controlador µDX100, que acessam nodos específicos e, por isso, não são contadas como blocos. É preciso usar um bloco **Expansão** para alocar as variáveis associadas as entradas e saídas das Expansões. Sem o bloco de **Expansão** no programa aplicativo este bloco de **Saída de Expansão** é inoperante.

ATENÇÃO: Como cada relé proporciona uma isolação galvânica suficiente, você pode conectar em seus contatos os fios da energia elétrica domiciliar, observando apenas o limite de corrente máximo de 10 Ampéres. Tome sempre muito cuidado nas conexões devido à proximidade entre

os contatos do conector.

Ao editar o bloco (aponte com o mouse para o bloco e pressione a tecla direita do mouse, ou pressione a tecla de espaço no teclado do computador) surge uma tela para inserção da saída e expansão a ser usada. Para o número da Expansão digite um valor entre 1 e 4 (desde que seja µDX100+, µDX100 permite apenas uma expansão), e para a saída um valor de 5 a 12:

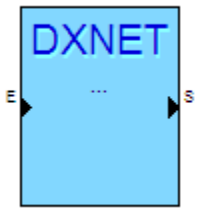


ATENÇÃO: No caso de uso de Expansões µDX110 é possível agregar até 8 expansões, sendo que cada Expansão possui 4 entradas e 4 saídas. Neste caso deve ser especificadas as entradas e saídas de 1 a 4, e não de 5 a 12, como no caso da Expansão do µDX100 (recurso disponível a partir da versão 2.0.2.0 do PG).

Veja também:
GERAL - Entrada de Expansão

GERAL - DXNET

Este é o bloco de instrução que permite a intercomunicação com vários µDX100 utilizando-se a Rede Local DXNET. Com ele é possível fazer duas operações: transferir para um nodo qualquer de outro µDX100 o estado atual do nodo de entrada deste bloco, ou transferir para uma variável qualquer de outro µDX100 o valor atual de uma variável local qualquer.

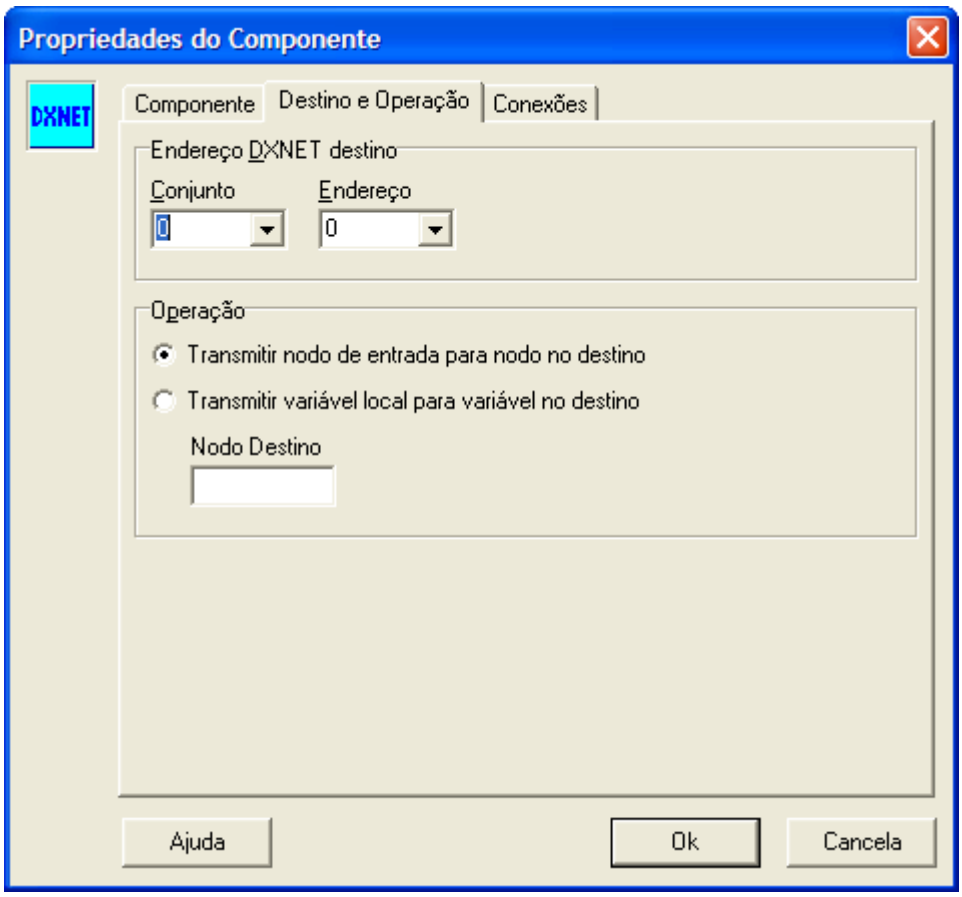


No caso de enviar variável a operação será realizada apenas quando o nodo de entrada passar do estado desligado para o ligado. Neste momento o µDX100 enviará o conteúdo da variável local para dentro da variável remota do µDX100 cujo endereço está especificado neste bloco de instrução (ou seja, apenas na borda de subida do sinal no nodo de entrada do bloco DXNET).

Se a instrução for para enviar nodo o estado atual do nodo de entrada será enviado a um rótulo (nodo) especificado no µDX100 de destino. Cada vez que o estado do nodo de entrada mudar, o µDX local vai transmitir ao µDX remoto o novo valor. O rótulo (nodo) no destino fica armazenado numa tabela especial onde existe uma posição (bit) para cada nodo do µDX. Esta tabela é sempre processada a cada ciclo e os estados nela contidos são reunidos com os estados de cada nodo do µDX destino, como se esta tabela estivesse representando as saídas de diversos blocos fantasmas. Esta tabela é chamada, internamente, de NODO_DX.

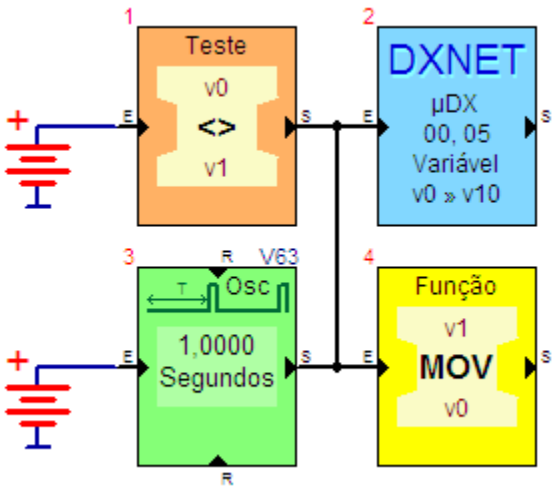
Em ambos os casos, envio de variável ou de nodo, o nodo de saída do bloco será ligado apenas se a comunicação tiver sido completada com êxito. Enquanto isto não ocorrer o µDX continuará tentando transmitir a informação a cada ciclo até que seja completada ou que o nodo de entrada volte ao estado anterior (como se não tivesse sido acionado).

Ao editar este bloco surge a seguinte tela:



O campo de endereço DXNET destino especifica o endereço do µDX100 que irá receber o nodo ou variável (conjunto e endereço DXNET), e o campo operação permite escolher entre

transmissão do nodo de entrada do bloco **DXNET** para o nodo especificado do µDX remoto, ou a escolha de uma variável local que será transmitida para uma variável do µDX100 remoto.

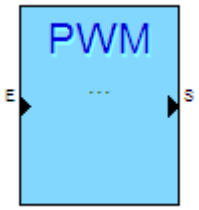


Exemplo de Programa Aplicativo: **Bloco_DXNET.d1g**

O programa de exemplo transmite a variável local v0 para a variável v10 do controlador endereço DXNET 5. Note que sempre que o valor de v0 muda ocorre a transmissão, já que neste caso v0 assume valor diferente de v1. Logo após o novo valor de v0 é movido para v1 e o bloco de comparação desliga. Para garantir que sempre v10 do µDX100 DXNET 5 esteja atualizada com o valor de v0 incluiu-se um bloco de oscilador com período de um segundo. Assim, mesmo que falhe a comunicação quando v0 muda de valor, a cada segundo v0 é transmitida.

GERAL - PWM

Este bloco de instrução permite que o µDX100 converta o período de um pulso repetitivo em uma das entradas em um valor de 8 bits para ser guardado em uma variável. Em última análise, este bloco permite fazer a leitura de sinais analógicos desde que se utilize, por exemplo, um circuito oscilador controlado por tensão conectado em uma das entradas (vide Entradas e Saídas).



Pode-se escolher uma de três entradas (E1, E2 ou E3) e qualquer das 64 variáveis do µDX100+ para armazenar o resultado (16 no caso de µDX100).

O bloco será ativado enquanto o nodo de entrada estiver ligado e o nodo de saída será ligado após a conversão. Se a largura do sinal da entrada em 1 ou em 0 exceder o limite de leitura o nodo de saída não será ligado. Os limites são:

Nível em 1 máximo: 1.215,23 µs

Nível em 0 máximo: 1.460 µs (mínimo de 10 µs)

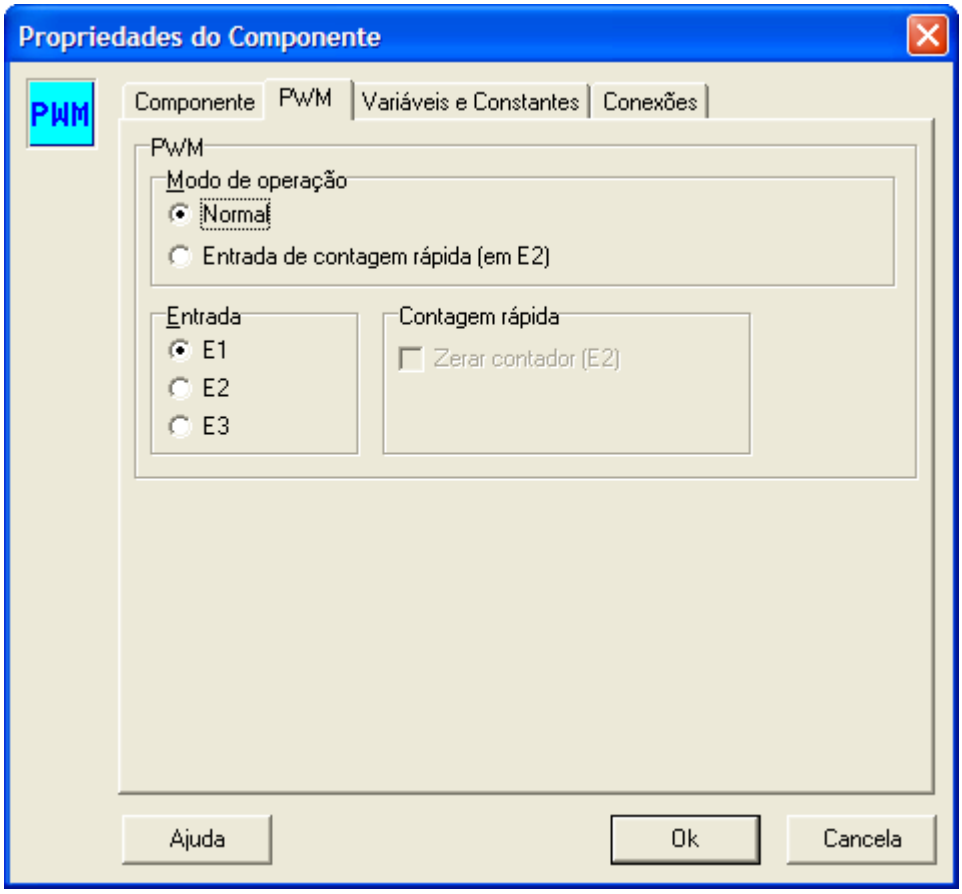
A medida de largura para a conversão é feita sobre o nível em 1 do sinal de entrada e está compreendida na seguinte faixa:

4,77 µs = 0

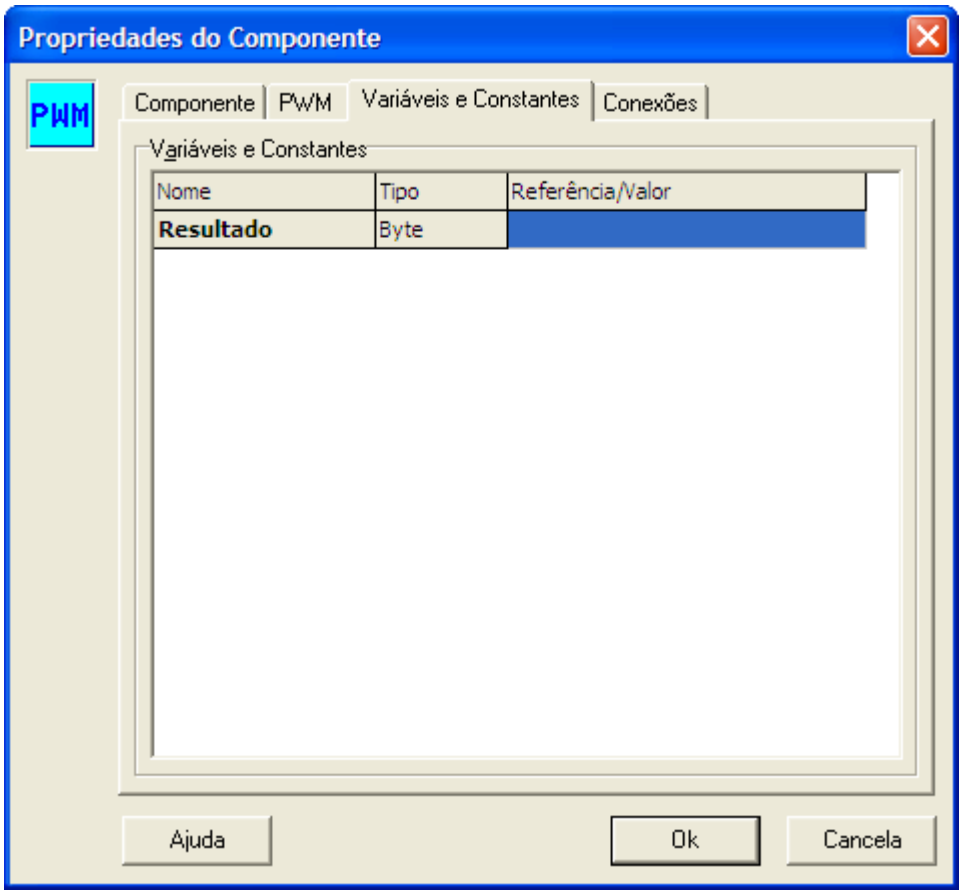
1215,23 µs = 254

A resolução da medida é de +/- 1, ou seja, um sinal em nível 1 com 14,3 µs de largura e que resultaria em uma medida correta de 3 pode ser medido como 2 ou como 4. Durações do nível 1 menores que 4,77µs poderão não ser detectadas, o que pode acabar causando ultrapassagem do limite do nível 0. Em µDX versão 4.2 em diante, caso a conversão PWM não ocorra corretamente, a variável associada ao PWM assume valor 255.

Ao editar este bloco surgem a seguinte janela:



No campo **Modo de Operação** é possível selecionar entre operação **normal** (leitura PWM, como descrito anteriormente) e **entrada de contagem rápida** (neste caso a entrada E2 permite ler pulsos até 3000Hz). Já o campo **Entrada** especifica qual das entradas do µDX100 será usada para leitura PWM (no caso de seleção de modo para contagem rápida a entrada é sempre E2). A aba **Variáveis e Constantes** permite escolher a variável que irá receber a conversão PWM:

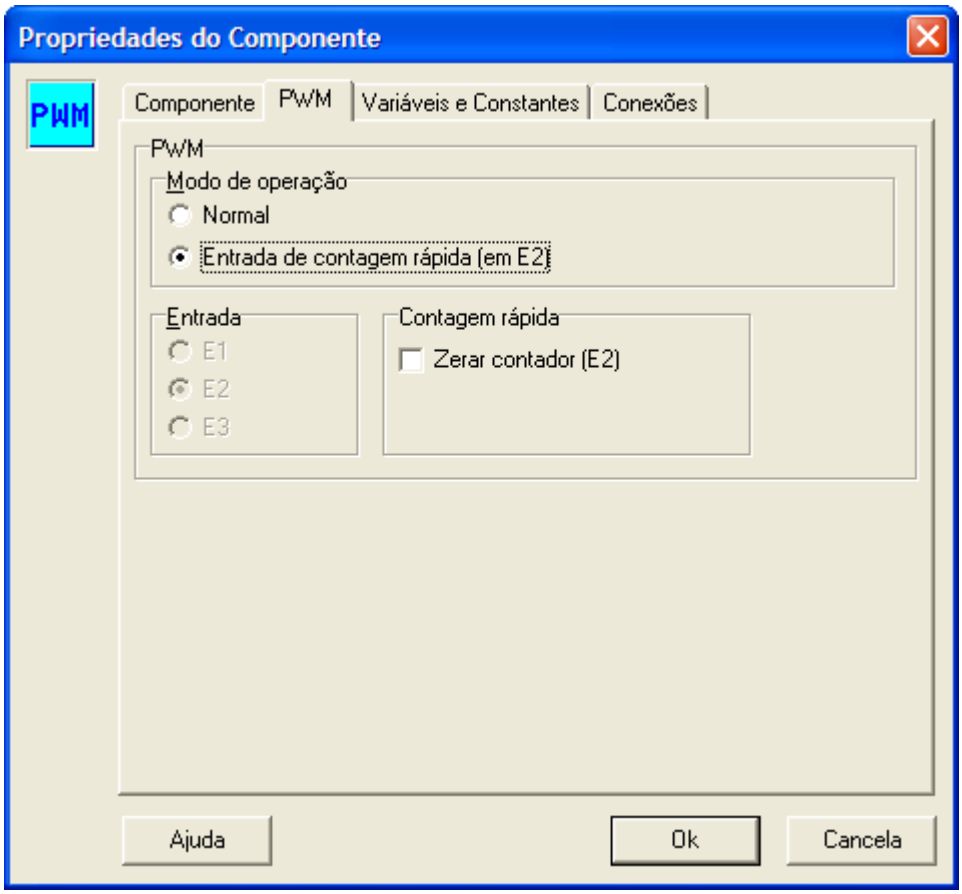

Propriedades do Componente
PWM
Componente
PWM
Variáveis e Constantes
Conexões
Variáveis e Constantes
Nome
Tipo
Referência/Valor
Resultado
Byte
Ajuda
Ok
Cancela

Entrada de Contagem Rápida

No caso de controlador µDX100+ ou µDX101 o bloco **PWM** pode ser usado para atribuir a entrada E2 um contador de alta velocidade, capaz de discernir até mais de 3000 Hz (3000 contagens por segundo). Note que a entrada E2 continua a se comportar como uma entrada normal, mas além disso a cada energização de E2 as variáveis de contagem rápida serão incrementadas.

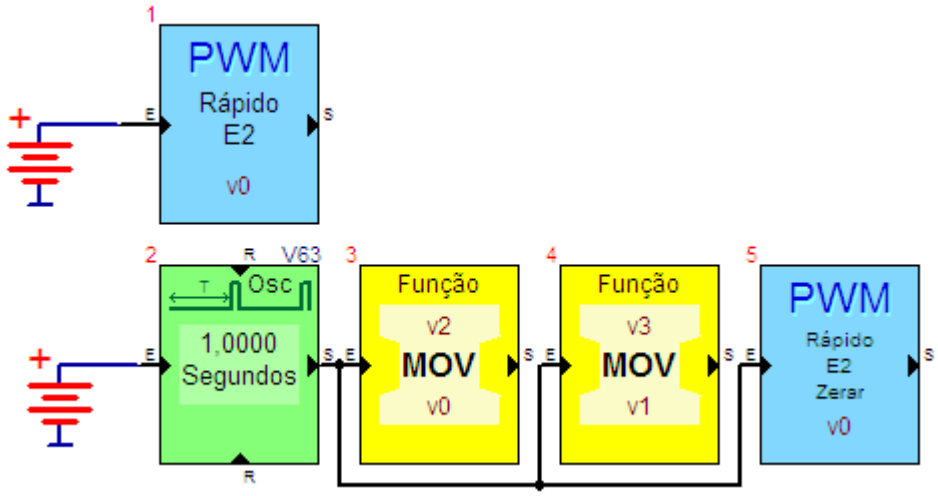
Importante: a opção de contagem rápida só é operacional para controlador programável µDX100+ ou µDX101. Em controladores µDX100 inexiste entradas de contagem rápida.

Ao selecionar a opção **Entrada de contagem rápida** o campo de **Entrada** é inibido (já que a contagem rápida só é efetuada no entrada E2 do µDX100+) e é ativada a opção **Zerar contador (E2)**:



Caso não seja selecionada a opção de **Zerar contador (E2)** este bloco irá incrementar a variável especificada e a subseqüente (de forma a formar um contador de 16 bits) conforme os pulsos na entrada E2.

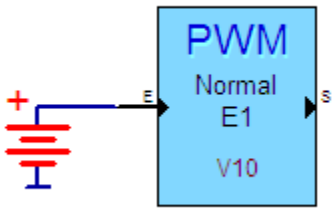
Já se for selecionada a opção **Zerar contador (E2)** este bloco irá zerar o contador rápido sempre que for energizado. Com isso, é fácil implementar, por exemplo, um freqüencímetro. Este exemplo é mostrado a seguir. Note que, a cada segundo, o valor das variáveis de contagem (v0, v1) são transferidas para outras variáveis (v2,v3), e a seguir são zeradas. Em v2,v3 irá surgir a freqüência presente na entrada E2, até o limite de 3000Hz. Veja que 1000Hz, por exemplo, será representado por 1000/256 = 3 e resto 232. Logo, para 1000Hz teremos v3=3 e v2=232. Note que a ordem dos blocos é importante. Assim, é preciso que os blocos que transferem a contagem para v2 e v3 (blocos 3 e 4) sejam anteriores ao bloco de zeramento da contagem (bloco 5).



Exemplo de Programa Aplicativo: **Bloco_PWM.d1g**

Entrada Analógica

No caso de controlador µDX101 o bloco **PWM** pode ser usado para leitura analógica da entrada E1. Note que a entrada E1 continua a se comportar como uma entrada normal, mas além disso ao usar o bloco PWM direcionado para a entrada E1 este permite ler a tensão existente nesta entrada com fundo de escala em 4,096V e resolução de 40mV (10 bits - 1024 steps). A variável indicada pelo bloco PWM recebe o byte LSB da conversão analógica, e a variável subseqüente recebe os dois bits MSB da conversão. Assim, ao usar um bloco PWM na entrada E1 com variável V10, por exemplo, a conversão usará a variável V10 para o valor LSB de conversão, e a variável V11 para o valor MSB de conversão analógica. Ao aplicar 0V na entrada E1 as variáveis V10 e V11 ficarão com valor zero. Já ao aplicar 4,096V as variáveis V10 e V11 assumirão valor 255 e 3, respectivamente. Isso corresponde ao valor de conversão 3 * 256 + 255 = 1023.



Importante: a opção de entrada analógica só é operacional para controlador programável µDX101. Em controladores µDX100 e µDX100+ inexiste entrada analógica.

Parte

Manutenção

O não funcionamento correto de qualquer uma das partes do Controlador Programável µDX Série 100 (seja o próprio µDX100 ou µDX101, o cabo de comunicação ou o programa) deverá ser comunicado diretamente à DEXTER.

Evite qualquer tentativa de conserto, adaptação ou configuração que não tenha sido cuidadosamente abordada neste manual.

A DEXTER não se responsabiliza pelo uso indevido ou incorreto do µDX100/µDX101 ou das partes que o acompanham.

Leia este manual com atenção antes de energizar o µDX100 ou µDX101.

Parte

Garantia

A DEXTER oferece uma garantia de 1 (um) ano, a contar da data da compra, para reposição ou conserto do todo ou das partes do Controlador Programável µDX100/µDX101 no caso de mau funcionamento ou defeitos originários na fábrica.

Esta garantia deixa de vigorar caso o defeito apresentado for resultante do uso indevido ou incorreto do todo ou das partes do µDX100/µDX101, assim como no caso de serem feitas alterações de qualquer espécie em qualquer das partes do µDX100/µDX101, sem autorização por escrito da DEXTER.

A DEXTER garante que o CD que acompanha o µDX100/µDX101 está isento de contaminação pelos vírus de computador conhecidos até a data de fabricação.

Não estão incluídos nesta garantia os custos com transporte do µDX100/µDX101 ou de suas partes, tanto para recebimento como para devolução.

Esta garantia se restringe ao controlador programável µDX100/µDX101, não se estendendo ao processo controlado, nem a sensores e/ou acionamentos ligados ao controlador. O bom funcionamento do µDX100/µDX101 pressupõe uma linha de alimentação sem ruídos e, no caso de acionamento de cargas indutivas, a instalação de supressores de ruído.

A DEXTER não se responsabiliza pela aplicação do µDX100/µDX101 em processos perigosos ou de risco de vida.

DEXTER Indústria e Comércio de Equipamentos Eletrônicos Ltda.
Av. Pernambuco, 1328, cjs. 307/309 - CEP:90240-001 - Porto Alegre - RS
Fone: (51) 3208-0533 - Celular: (51) 99963-0370
Página Internet: http://www.dexter.ind.br
E-mail: dexter@dexter.ind.brbr

Índice

- 1 -

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- I -

- L -

- M -

- N -

- O -

- P -

- R -

- S -

- T -

- V -

- Z -