CONTROLADOR PROGRAMÁVEL
GP3320

Instalação e Características Técnicas

Versão 2022-1
Ref.: 31940522-7

BCM automação

ATENÇÃO

Os Controladores Programáveis BCM são equipamentos robustos e confiáveis. O seu projeto foi feito levando em conta as condições de operação particulares do ambiente industrial. Porém, nunca esqueça que todos os elementos de um sistema estão sujeitos a falhas. Desenvolva o projeto do sistema levando isto em consideração, obedecendo rigorosamente as recomendações deste manual e as normas de segurança vigentes em seu país ou região.

Um bom projeto do sistema e uma correta instalação são elementos fundamentais para o funcionamento satisfatório e confiável dos produtos.

Caso haja qualquer ponto duvidoso ou omisso, não hesite em consultar o Departamento de Assistência Técnica da BCM, o qual terá o maior prazer de lhe prestar todo o apoio necessário.

Telefone: (51)3374.3899
E-mail: sistemas@bcmautomacao.com.br
www.bcmautomacao.com.br

ÍNDICE GERAL

1. INTRODUÇÃO

Este capítulo apresenta os conceitos básicos associados às características e à aplicação dos Controladores Programáveis.

1.1 O que é um Controlador Programável

O Controlador Programável é o equipamento mais importante em uso na automação de equipamentos e processos industriais no Brasil e em todos os países desenvolvidos. Seu campo de aplicação é quase ilimitado e o conhecimento de suas potencialidades torna-se cada vez mais necessário a todos os profissionais envolvidos no planejamento, operação e manutenção de processos industriais.

O Controlador Programável (CP) é um equipamento eletrônico programável baseado em microprocessadores. É projetado para funcionar em ambientes industriais, podendo controlar desde simples máquinas e processos até automatizar uma planta completa.

Este manual contém a documentação de aplicação dos módulos e os procedimentos usuais que devem ser executados numa instalação típica dos Controladores Programáveis (CPs) da linha GP3320. É importante que o técnico encarregado da instalação tenha uma visão de conjunto do Controlador Programável, que é a proposta deste primeiro capítulo.

1.2 Estrutura do Controlador Programável e Modo de Operação

O Controlador Programável possui três blocos básicos: as Entradas, a Unidade Central de Processamento (CPU) e as Saídas. Através de dispositivos ligados ao Módulo de Entradas, o CP monitora continuamente o estado da máquina (ou processo) sob seu controle. A Unidade Central de Processamento processa os dados externos através do Programa do Usuário (Programa de Controle gravado previamente na memória do CP). Simultaneamente, as saídas são acionadas conforme instruções contidas no mesmo programa.

Desta forma, um CP *sente, decide e age* sobre a máquina (ou processo) conforme uma lógica preestabelecida. Os dispositivos de Entrada e Saída podem ser das mais diversas categorias e tipos, tanto analógicos quanto digitais, em faixas de tensão e corrente as mais diversas:

- **Botoeiras;**
- **Transdutores;**
- **Chaves fim de curso;**
- **Motores;**
- **Contatores;**
- **Solenoides;**
- **Alarmes sonoros e visuais;**
- **Pressostatos, termostatos;**
- **Instrumentos analógicos (termopares, etc.);**

1.3 Aplicações dos Controladores Programáveis

O Controlador Programável é um equipamento extremamente versátil, com aplicações em todos os segmentos industriais. Suas características permitem que ele efetue desde simples lógicas até sofisticados controles de processos. Atualmente, existem modelos de CPs que permitem, de maneira econômica, controlar mecanismos e processos a partir de poucos pontos de entrada e saída. Sistemas que utilizam lógica pneumática ou de relés comportam a substituição direta dos circuitos lógicos por um CP, com vantagens imediatas em termos de confiabilidade, facilidade de manutenção, ocupação de menor espaço físico, diminuição do peso e versatilidade em nível de futuras alterações ou aperfeiçoamento da lógica de controle.

Máquinas ou processos que requeiram o controle simultâneo de variáveis em diversos pontos, exigindo relações complexas de controle em um ou mais pontos do processo ou mesmo em outras máquinas, adaptam-se muito bem ao uso com CPs, pois estes permitem a leitura de variáveis analógicas e digitais, o processamento rápido das informações e a geração de sinais de saída analógica ou digitais.

Todos os modelos de CPs BCM possuem canais de comunicação que permitem a conexão de um controlador a outro ou a um computador central. Esta possibilidade abre um campo totalmente novo: um computador central pode monitorar a operação dos CPs, verificando anomalias, detectando falhas na produção, emitindo relatórios, etc., ao mesmo tempo em que pode interferir na operação do CP, modificando parâmetros, iniciando ou interrompendo sequências em função de um planejamento global da planta industrial ou de fatos ocorridos em outros processos. É importante ressaltar que os Controladores Programáveis não são apenas substitutos mais confiáveis do que os relés. Na verdade, eles representam um salto qualitativo em termos de controle, pois viabilizam soluções inovadoras nos processos e automatismos onde são empregados, resultando em consideráveis incrementos na eficiência dos mesmos.

A seguir, relacionamos alguns exemplos de máquinas e processos que podem utilizar, sendo impossível esgotar todas as aplicações. Pode-se afirmar que praticamente qualquer máquina ou processo que possua alguma lógica de controle, pode utilizar um CP.

AUTOMAÇÃO DE MÁQUINAS	CONTROLE DE PROCESSOS
Injetoras de plástico	Metalúrgicos
Extrusoras	Siderúrgicos
Prensas	Químicos
Furadeiras	Medição e controle de energia
Prensas e retíficas	Estufas e secadoras
Plainas	Supervisão de plantas industriais
Máquinas impressoras	Sistema de transporte e armazenamento
Esmerilhadoras	
Robôs e manipuladores	
Misturadores	
Câmaras de vácuo	
Bobinadoras de motores	
Máquinas especiais	
Automatismos em geral	

2. DIRETRIZES DE SEGURANÇA E INSTALAÇÃO

Este capítulo apresenta os requisitos e orientação para uma instalação correta, adequada e segura dos Controladores Programáveis GP3320.

2.1 Segurança

Sempre que alguma ação de máquinas ou processos sob controle do GP3320 possa causar danos a um ser humano ou prejuízos de quaisquer ordem, DEVEM ser previstas redundâncias mecânicas e elétricas INDEPENDENTES do controlador, de modo a garantir a segurança do sistema numa eventual falha do controlador ou em eventuais erros de sua programação.

O Controlador Programável NÃO deve ser aplicado em sistemas dos quais dependam a vida de seres humanos, a menos que sejam previstas seguranças e redundâncias que evitem acidentes causados por eventuais defeitos ou falhas.

As normas de Segurança vigentes no país ou região onde será utilizado o Controlador Programável devem ser seguidas rigorosamente. No caso de alguma destas normas conflitar com as recomendações de instalação e uso do Controlador GP3320, o usuário do CP deve fazer uma comunicação por escrito a BCM e ficar aguardando que a BCM dê uma posição para o caso.

Devido à diversidade de aplicações e usos destes equipamentos e também as suas particularidades enquanto equipamentos eletrônicos, a BCM não se responsabiliza por danos indiretos ou diretos ocasionados pela utilização do CP.

O sistema mecânico, elétrico e a programação devem ser projetados de modo que, em caso de falta de energia, o sistema ofereça segurança ao operador, não provocando movimentos ou ações danosas, tanto no momento da falta quanto no retorno da energia.

ATENÇÃO!

Não esqueça que o CP, apesar de ser um equipamento robusto e confiável, também é sujeito a defeitos ou mau funcionamento. Um bom projeto de sistemas deve levar em conta estas considerações SEMPRE.

2.2. Instalação

Antes de ser iniciado o trabalho de montagem e instalação, este manual deve ser lido atentamente. Aqui estão relacionados os procedimentos e cuidados que usualmente devem ser levados em conta numa instalação típica do Controlador Programável GP3320.

2.2.1 Inspeção e Pré-montagem

a) Certifique-se de que o aparelho corresponde exatamente ao pedido feito, contendo todos os módulos, fontes de alimentação, displays, teclados e cabos especificados no pedido. Caso falte algum dos itens, entre imediatamente em contato com a BCM.

b) Certifique-se que a tensão de alimentação dos módulos corresponde à especificada no pedido e seja compatível com a aplicação.

c) Faça um exame visual cuidadoso de todos os componentes do controlador programável. Qualquer dano causado por transporte deve ser comunicado imediatamente ao transportador e à BCM.

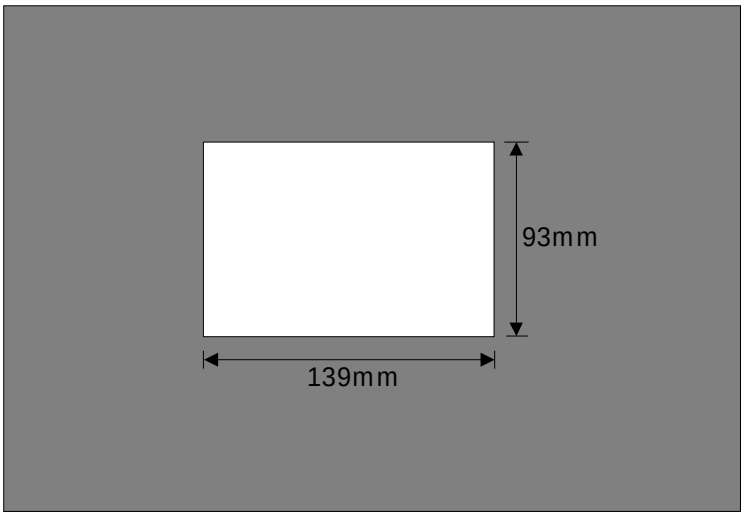
A partir deste momento, o controlador está pronto para ser instalado. A instalação começa pela fixação do CP no gabinete apropriado.

ATENÇÃO!

Desconecte o Controlador da Rede de Alimentação sempre que houver necessidade de manipular qualquer de seus elementos. Isto deve ser observado em todas as situações.

2.2.2. Montagem Mecânica

O Controlador Programável deve ser instalado em um gabinete que possua vedação completa contra poeira, respingos de água, óleo e produtos corrosivos. Este gabinete também deve protegê-lo contra choques mecânicos, vibrações mecânicas e altas temperaturas (acima de 45 °C no ambiente externo ao gabinete).



Corte no painel para fixação frontal

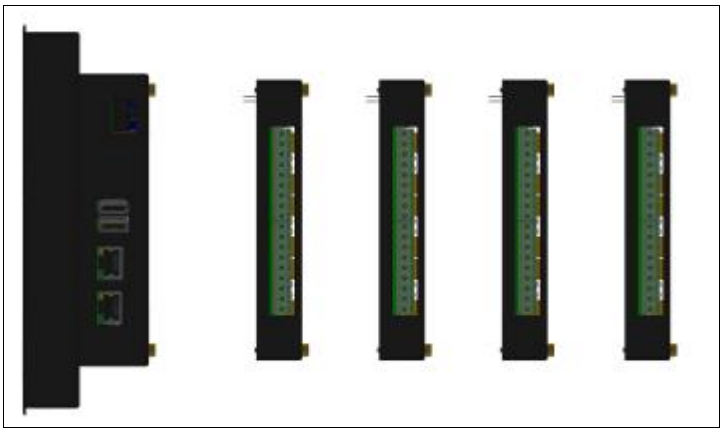
A profundidade do controlador varia entre 65 e 200 mm, conforme o tipo e o número de módulos de E/S do controlador.

Ao planejar a instalação, deve ser prevista uma folga mínima de 75 mm acima e abaixo do controlador, para ventilação, acesso aos conectores e passagem da fiação.

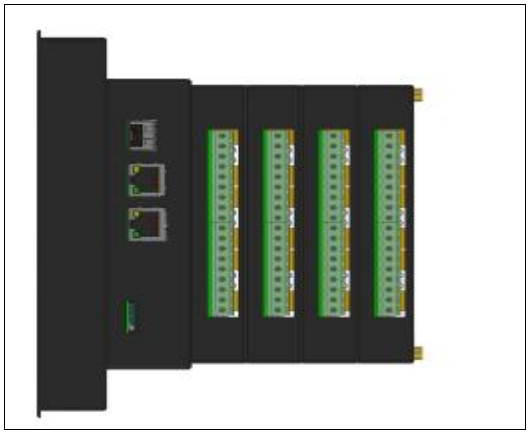
.

2.2.3. Instalação dos módulos no barramento local

O GP3320 suporta até 4 módulos de E/S instalados no barramento local localizado no painel traseiro do módulo base, instalados conforme as figuras abaixo:



Vista explodida do GP3320 com os módulos de E/S



Vista do GP3320 com os módulos montados

2.2.4. Temperatura de Funcionamento

Em que pese os controladores GP3320 terem sido projetados para operar em temperatura ambiente de até 55°C, deve-se ter em vista que a vida útil dos componentes eletrônicos diminui com o aumento de temperatura. Portanto, é conveniente que o equipamento opere em temperaturas próximas de 25°C. Para que esta condição seja mantida, deve ser prevista uma ventilação eficiente, caso o ambiente externo ao gabinete onde está instalado o CP tenha temperaturas acima de 45°C. Esta ventilação deve manter a condição de "Vedação completa contra poeira, respingos de água, óleo ou produtos corrosivos".

ATENÇÃO!

O controlador é projetado de modo que o calor gerado nas placas seja dispersado por convecção – o que é suficiente na maioria dos casos. Por este motivo, deve ele ser instalado na horizontal, observando os espaços de ventilação.

2.3. Normas e cuidados para instalação da fiação

2.3.1. Aterramento

Os Controladores da linha GP3320 possuem um terminal GND que deverá ser conectado ao terminal ou borne de aterramento da instalação, garantindo a equipotencialidade das estruturas dos equipamentos. Este terminal não deverá ser ligado ao neutro da rede. Recomendamos que o aterramento seja sempre feito de acordo com as prescrições da norma ABNT-NBR5410.

2.3.2. Cablagem

Uma fiação limpa e bem instalada é fundamental para o bom funcionamento do sistema. Tenha em mente que a própria fiação faz parte do sistema.

A seguir, estão relacionados alguns procedimentos gerais que uma vez obedecidos, aumentarão a confiabilidade do sistema.

Seguir, rigorosamente, o projeto e documentação referentes à instalação das fiações.

a) Os seguintes itens devem ser conferidos:
- Terminais e conectores (quantidade e espécie).
- Numeração dos fios e bornes.
- Distribuição física dos fios e cabos. Esta distribuição deve estar de acordo com as normas e recomendações contidas neste manual, a fim de que seja evitado o ruído elétrico ou outras possíveis consequências de uma instalação imperfeita.

b) Utilizar cores diferentes para as fiações de entrada e de saída.

c) Utilizar cores diferentes para os fios que conduzem sinais analógicos e para os fios que conduzem sinais digitais.

d) Usar cabos blindados nas ligações que conduzem sinais analógicos.

e) Descascar as pontas dos fios e cabos apenas com o Alicate de Prensa Terminal (tipo AMP, Burndy, etc.). Este alicate é o único tipo de ferramenta que propicia o correto desencapamento das ponteiras de fios e cabos. Nunca improvise outro tipo de ferramenta, pois isto pode provocar rupturas internas nos cabos.

f) Utilizar, sempre, terminais de boa qualidade. A fixação das pontas dos fios aos conectores, sempre deve ser feita através de terminais.

Observação:
Nunca estanhar as pontas dos fios. Embora a estanhagem dê a aparência de uma boa fixação, com o tempo é provável que o fio se solte do borne.

2.3.3. Ruído Elétrico

Os ruídos elétricos são distúrbios que ocorrem na alimentação do sistema, normalmente de tensão e frequência elevadas. No ambiente industrial, estes fenômenos são mais frequentes e mais intensos, devido ao grande número de motores e outras cargas sendo acionadas e desativadas frequentemente. Estes distúrbios podem provocar falhas de funcionamento nos circuitos eletrônicos. O GP3320 foi projetado com proteções, cuja finalidade é evitar ao máximo a interferência do ruído. Porém, a maior segurança possível é conseguida eliminando-se as fontes de ruído existentes e cuidando em evitar os possíveis acoplamentos entre o CP e os prováveis caminhos de ruído elétrico. Para isto, é fundamental o cuidado com a fiação e lay-out do sistema (planejamento do sistema). Não devem ser montados no mesmo painel do Controlador Programável: transformadores, contatores, solenoides ou outros elementos eletromecânicos não concernentes ao Controlador.

O ruído elétrico pode se propagar de forma irradiada, indutiva, capacitiva ou condutiva. O controlador programável BCM foi projetado para minimizar estes acoplamentos.

Porém, em que pese todos os cuidados no projeto do CP, uma instalação "limpa" e cuidadosa permite evitar possíveis problemas. Para tanto, a seguir estão relacionados os cuidados que devem ser tomados na fiação do sistema e esquemas para a supressão de ruídos. Para efeitos de ruídos, podemos dividir a fiação do Controlador Programável em:

Nível 1: Entradas e saídas analógicas
Cabos para displays, teclados ou outros bastidores
Comunicação através do canal RS232

Nível 2: Entradas digitais 24 Vcc
Saídas digitais 24 Vcc
Comunicação entre sistemas no canal RS485 ou Ethernet

Nível 3: Entradas Digitais 110 ou 220 Vac
Saídas digitais 110 ou 220 Vac

Nível 4: Rede de alimentação e aterramento

Fiações de diferentes níveis não devem ser misturadas – A distribuição de bornes nos armários, caixas de passagem, painéis, etc. deve levar em conta a separação entre os níveis. Da mesma forma, devem existir conduítes e eletrodutos separados para a fiação correspondente a cada nível.

2.3.4. Fusíveis

Recomendamos a instalação de fusíveis para proteção dos circuitos dos módulos de saída do CP.

A não utilização de fusíveis poderá acarretar danos permanentes nos circuitos de saída do CP, caso aconteça um curto-circuito nas fiações ou cargas. Os fusíveis podem ser instalados de dois modos:

- Instalação de fusíveis individuais em cada saída. Neste caso, devem ser utilizados fusíveis do tipo rápido, com capacidade de ruptura de 2A/250 V.

– Um único fusível num ponto comum de alimentação do cartão. Neste caso, deve ser utilizado um fusível com capacidade de ruptura de 6A/250 V.

2.4. Partida do Sistema

2.4.1. Pré-Teste

Após montar e instalar o Controlador Programável conforme as normas e indicações que constam do Manual de Instalação, deve ser feito um Pré-teste *antes* do sistema entrar em funcionamento efetivo. O sistema compreende o CP, as Interfaces, Fiações, Botoeiras, Sensores, Estabilizador de Tensão (quando houver) e a Máquina ou Processo ao qual o CP está acoplado. O Pré-Teste compreende os seguintes passos:

Verificação Preliminar:

a) Verificar que o controlador esta desconectado da rede elétrica

b) Verificar se a tensão de alimentação corresponde ao especificado (115/220Vac, 12Vcc ou 24Vcc, conforme o modelo de fonte).

c) Verificar se as tensões das entradas e saídas correspondem as especificadas no manual. Isto deve ser feito antes do conector ser encaixado na placa respectiva.

Verificação das entradas:

a) Usando a listagem de entradas, verificar se cada sensor, botão e cada chave corresponde efetivamente a sua respectiva entrada, descrita no programa.

b)As falhas possíveis no módulo de entradas podem estar vinculadas a:

- **Fonte de alimentação;**
- **Erro de fiação (trocada, partida, mal conectada);**
- **Conexão das entradas;**
- **Dispositivo de entrada (sensor mal posicionado ou com defeito, etc.)**
- **Defeito no próprio módulo de entradas;**

ATENÇÃO!

Não inserir, em hipótese alguma, ponteiras de teste ou outro instrumento nos conectores. Este procedimento pode danificar o mesmo, provocando defeitos intermitentes.

. **Verificação das Saídas:**

a) Usando a Listagem das Saídas, verificar se cada saída corresponde à respectiva carga descrita no programa.

b) Antes de ligar a alimentação do CP pela primeira vez, desligue todas as válvulas solenoides, os motores de seus contatores e quaisquer outros elementos que possam causar movimentos perigosos na máquina ou processo, capazes de provocar danos materiais e acidentes pessoais.

c) Uma outra forma de desativar os movimentos perigosos é desligar a pressão de óleo e ar de todos os mecanismos que possam provocar movimento da máquina. Desligar também da rede de potência todos os motores e outros dispositivos eletromecânicos que possam causar ações indevidas.

2.4.2. Teste final do sistema

O próximo passo é verificar os movimentos de máquina (ou etapas do processo) controlados pelo programa do CP. Para isto, deve ser feito um Teste Sequencial, de acordo com o Diagrama de Estados da máquina ou processo a ser controlado.
A mudança de Estado só ocorre se forem satisfeitas as condições de mudança de estado. Os estados e as condições de mudança estão descritos no diagrama de estados da máquina.
Para o Teste Final do Sistema, deve ser adotado o seguinte procedimento, que evoluirá do estado ZERO (máquina desligada) até o último, passando portanto por todos os estados. Ao final, teremos executado o ciclo completo da máquina a ser controlado.

ATENÇÃO!

Nesta etapa, alguém deve estar próximo a uma chave de parada de Emergência, ou de desligamento do sistema, e pronto a acioná-la em caso de necessidade. Certifique-se, antes de iniciar esta etapa de teste, de que nenhum dispositivo de saída permaneça acionado por qualquer causa. Verifique também que nenhum movimento ou acionamento indesejado possa ocasionar danos.

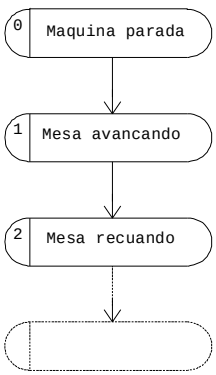
Roteiro para o teste final:

a) Com o Diagrama de Estados em mãos, devem ser verificadas as condições de transição.

b) Com o Sistema desconectado da alimentação, devem ser ligados os dispositivos de Entrada e Saída que permitirão ao Sistema sair do estado ZERO e passar ao estado UM. Todos os outros dispositivos de E/S devem permanecer desligados.

c) Acione o botão (ou chave) de INÍCIO e verifique se o Sistema passa do Estado ZERO para o Estado UM dentro das condições estabelecidas para mudança de Estado.

Exemplo: Vamos supor que o Estado UM de uma máquina seja o de, logo após o acionamento do botão (ou chave) de início, avanço de uma mesa. Quando a chave fim de curso é acionada, o CP passa para o estado DOIS, que será o recuo da mesa. A primeira parte do diagrama de Estados estará representada assim:



Neste caso, o procedimento de teste (Estado 1) deve ser iniciado da seguinte forma:

a) Desligar a chave fim de curso. Isto impedirá que a máquina passe do estado UM para o estado DOIS;

b) Acionar o botão de início (ou a chave). A mesa deve avançar;

c) Desconectar a alimentação do sistema, para que a mesa não continue avançando.

d) Desta forma, o Estado 1 da máquina esta testado. Agora, passamos para a etapa seguinte, que é teste do Estado 2. Para isso, fazemos a máquina voltar ao estado ZERO (mesa recuada e alimentação desconectada), fechando a chave fim de curso.

Procedimento para o teste do estado DOIS:

a) Fazer a máquina retornar ao estado ZERO;

b) Conectar a chave fim de curso;

c) Desconectar os dispositivos que fariam a máquina passar ao estado seguinte (Estado 3).

Assim, a máquina sai do estado ZERO, passa pelo estado UM (já testado) e vai ao estado DOIS. A partir dai, já se pode iniciar o teste do estado TRÊS.

Devemos continuar testando o sistema estado a estado, num procedimento evolutivo e sequencial. Ao final, a máquina terá cumprido todo seu ciclo de trabalho e estará testada. A partir daí, o sistema já poderá entrar em operação efetiva.

Caso algum módulo apresente problema ou algum estado de máquina e suas condições de saída não estejam de acordo com o Diagrama de Estados da máquina, siga o roteiro para busca de falhas. Antes, porém, certifique-se de que o Diagrama de Estados corresponde ao ciclo de trabalho real do processo a ser controlado. Verifique, também, a possibilidade de erros de programação. Para isto, compare a listagem do programa de controle com o Diagrama de Estados.

3. CARACTERÍSTICAS DOS CONTROLADORES PROGRAMÁVEIS GP3320

A série GP3320 de controladores programáveis possui uma estrutura modular, compacta e totalmente preparada para IoT e Indústria 4.0, abrangendo um vasto leque de configurações possíveis de produtos.

O módulo básico, incluindo o IHM, CPU e fonte de alimentação, é projetado para fixação frontal, permitindo a instalação de até 4 módulos de entrada/saída do padrão GP3000, suportando também até 1024 módulos de entrada/saída remotos usando o barramento de campo distribuído EtherCAT.

A programação dos produtos GP3320 é feita através da ferramenta Solver, um avançado ambiente de programação multiplataforma compatível com a norma IEC61131-3.

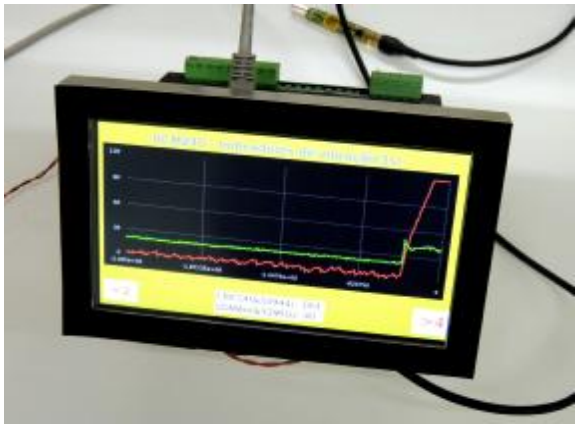
3.1. Características gerais da série GP3320

- ***Até 61560 pontos de E/S;***
- ***Multibarramento de dados;***
- ***Cinco canais de comunicação: Ethernet Elétrico/Óptico 1G/100Mbps, RS232, RS485 e USB;***
- ***IHM LCD de 7", touch screen, alta resolução e aceleração gráfica em hardware;***
- ***Sistema operacional BCM Real Time OS, multitarefa, com escalonamento preemptivo;***
- ***Muitos protocolos disponíveis (MODBUS, DNP, IEC 61850, etc);***
- ***Programação de acordo a norma IEC61131-3;***
- ***Operações em ponto flutuante, funções matemáticas avançadas, matrizes multidimensionais;***
- ***Segurança – Controle de acesso de usuários via username e password;***
- ***Servidor web embarcado para configuração via web browser;***
- ***Atualização de software segura sem parada do programa do usuário (on the fly);***

3.2. O controlador programável GP3320

O módulo básico GP3320 oferece:

- **IHM gráfico colorido touchscreen**
- **Dois canais seriais, um RS232 e um RS485**
- **Dois canais Ethernet, 1GB/s e 100MB/s**
- **Duas portas USB**
- **Cartão memória SD**
- **Cage para interface óptica;**
- **Barramento local BCM;**
- **Barramento distribuído EtherCAT;**
- **Alimentação 24Vdc;**



3.2.1. Versões disponíveis

Referência	Descrição
31021980-0	GP3320/CMP7 – Módulo básico com display gráfico 7", touchscreen, dois canais Ethernet e barramento EtherCAT
31021982-4	GP3320/SDP - Módulo básico sem display, demais recursos similares ao /CMP7
31021981-2	GP3320/ULC1 - Módulo básico com recursos reduzidos em relação ao /CMP7 (sem suporte ao barramento Ethercat e sem o segundo canal Ethernet), com display gráfico 7" e touchscreen
31021983-6	GP3320/ULC2 - Módulo básico com recursos reduzidos em relação ao /CMP7 (sem display, sem suporte ao barramento Ethercat e sem o segundo canal Ethernet)
31021990-3	GP3ETK - Módulo Ethercat para suporte a entradas/saídas remotas, com capacidade de suporte de até 4 módulos de E/S padrão GP3000

3.2.2. Escolha dos módulos de entrada/saída

Fazer um levantamento da necessidade de pontos de entrada/saída da aplicação - Número de pontos e tipo de sinais.
Escolher a combinação de módulos que atenda as exigências, considerando que o módulo básico suporta até 4 módulos de E/S. Os diferentes tipos de módutos de E/S disponíveis são:

GP3ESD	Entradas/saídas digitais
GP3ESA	Entradas/saídas analógicas, mais entrada para encoder
GP3ESH	Entradas/saídas digitais e analógicas no mesmo módulo Versões específicas para entrada de sensores Pt100
GP3VAF37	Entradas/saídas digitais e analógicas, mais entradas para tensão e corrente AC para medida de grandezas elétricas (tensão, corrente, potência, etc.)

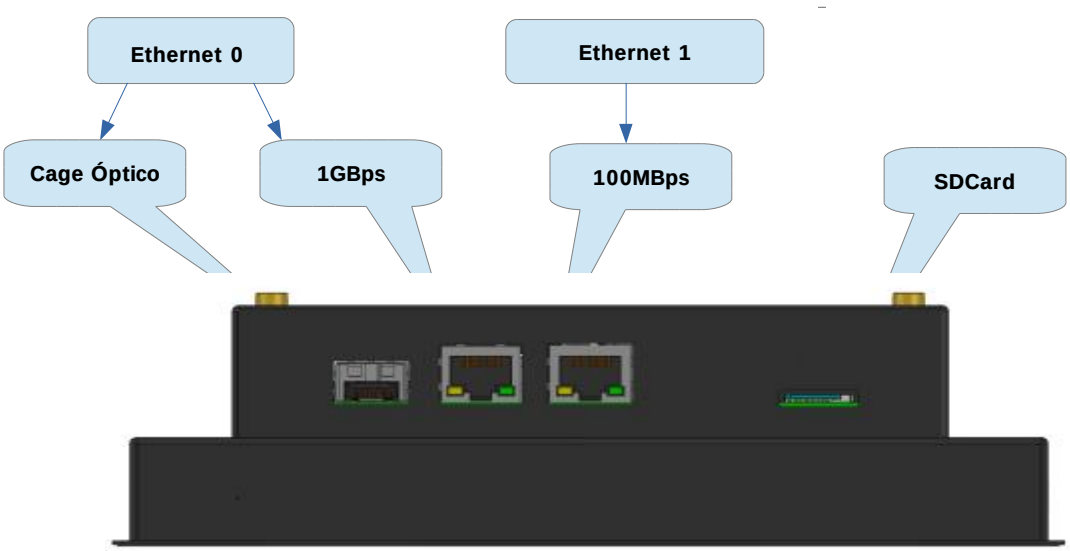
Consulte o manual GP3000.PDF para detalhes e informações completas sobre as características e referências dos produtos disponíveis para cada tipo de módulo.

3.2.3. Características técnicas detalhadas do GP3320

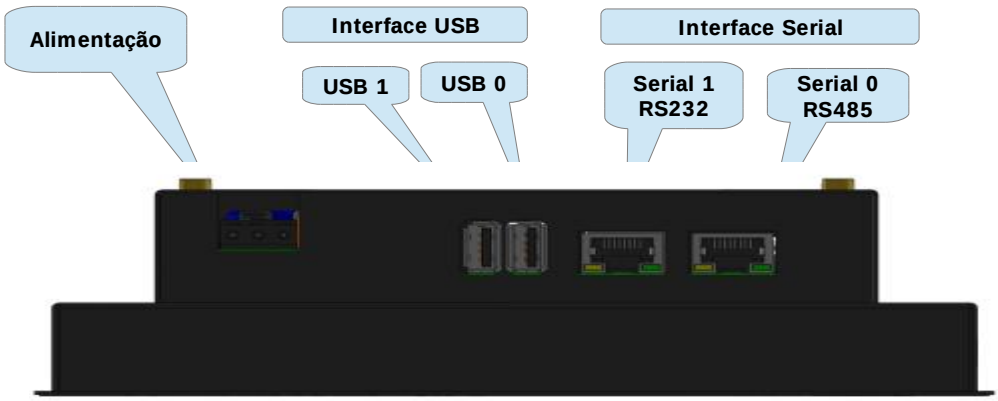
Sistema Operacional	**BCM Real Time OS**
Capacidade de memória	**256 MBytes memória não volátil** **512 MBytes memória de dados** **- Compartilhadas entre o Sistema Operacional e o programa do usuário** **64 KBytes memória retentiva** **- Utilizada somente para persistência das variáveis do programa do usuário**
Programa de usuário conforme IEC61131-3	**- Até 65536 tarefas com 100 níveis de prioridade e periodicidade configurável, com escalonamento preemptivo** **- Número de PROGRAMS, FUNCTIONS e FUNCTION BLOCKS limitado somente pela memória do sistema** **- Programação estruturada e modular, oferecendo as linguagens Ladder (LD), Structured Text (ST), Sequential Function Chart (SFC), Function Block Diagram (FBD), Instruction List (IL) e Linguagem Descritiva BCM** **- Múltiplos tipos de dados - Inteiros de 8 até 64 bits, ponto flutuante de 32 e 64 bits, arrays, variáveis tipo TIME (data e hora)** **- Conjunto abrangente de funções matemáticas**
Protocolos de comunicação	**MODBUS TCP Server/Client** **MODBUS RTU Master/Slave** **EtherCAT** **OPC-UA** **IEC 61850**
Programação e configuração	**Via ferramenta Solver e interface Web**
Barramento de E/S	**- BCM BUS (Capacidade para até 4 módulos padrão GP3000)** **- EtherCAT**
Interfaces de comunicação	**1 x 1Gbps Ethernet Elétrico/Óptico (Combo)** **1 x 100Mbps Ethernet Elétrico** **1 x RS232 (1,2 a 115,2kbps)** **1 x RS485 Isolada (1,2 a 115,2kbps)** **2 portas USB**
Módulos Ethernet Ópticos	**1Gbps ou 100Mbps padrão SFP para operação bidirecional (par de fibras) ou unidirecional (mono fibra) com opções de curto ou longo alcance (>10Km). Consulte a BCM ENGENHARIA para verificar os modelos disponíveis.**
Data e Hora	**Relógio em tempo real (RTC)**
Armazenamento externo	**Cartão de memória padrão MicroSD (4 a 32GB)**
IHM	**Display touchscreen colorido de 7 polegadas (800 X 480 pixels)**
Alarme sonoro	**Com frequência e duração configuráveis**
Alimentação	**24Vcc nominal (22 a 30V), 300mA (só módulo básico), máximo de 1A com todos os módulos de E/S**
Segurança	**Controle de acesso de usuários via username e password**
Atualização de firmware	**Atualização segura disponível via FTP**
Temperatura de Operação	**-20 a +70°C**
Relógio / calendário	**Resolução de 1s. Erro máximo de 100ppm (9 segundos/dia)**
Persistência do relógio e memória retentiva	**Retenção para um período de até 40h sem alimentação**
Dimensões	**Frontal: 185 x 120mm** **Corpo: 140 x 94mm** **Área de display: 153 x 85mm** **Profundidade: 60mm (só o módulo básico)**
Peso	**600g**

3.2.4. Conexões externas

3.2.4.1 Vista superior



3.2.4.2 Vista inferior



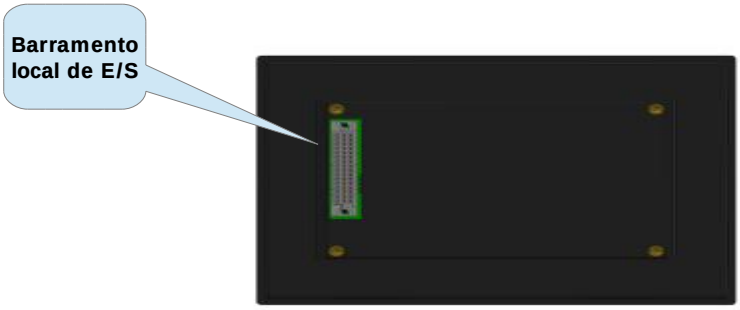
Conector Serial 0 (RS485)

Pino	Sinal
4	VL-
5	S-
6	S+

Conector Serial 1 (RS232)

Pino	Sinal
2	RxD
3	TxD
5	GND
6	Saída 5V

3.2.4.3. Vista traseira



3.2.5. Configuração do GP3320 via página Web

Os controladores programáveis GP3320 vem configurados de fábrica com o IP: 192.168.50.212. Caso seja necessário, o usuário deve acessar este endereço através de um web browser para alterar os parâmetros. As seguintes configurações são realizadas através da página Web:

- **Endereço IP**
- **Velocidade, paridade e controle de fluxo dos canais seriais RS232 e RS485**
- **Data e hora**

Antes de entrar na tela de configuração, o GP3320 pede o seu login e senha. Na versão atual, estes estão fixos:

Login: admin
Senha: bcmbcm

Após efetuar a modificação de qualquer um dos parâmetros anteriormente citados, haverá a necessidade de realizar a reinicialização do CLP de modo que a atualização dos parâmetros seja de fato efetivada. Toda e qualquer alteração dos parâmetros citados é gravada de forma que a configuração não será perdida na falta de alimentação.

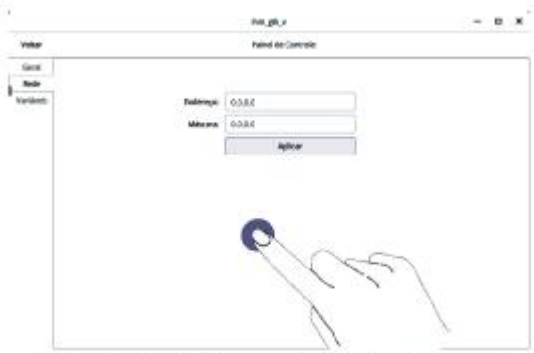
Tela de configuração dos parâmetros de endereço IP e interfaces seriais

Seção da tela de configuração responsável pela atualização de data e hora

3.2.6. O display de serviço

Se por qualquer motivo o usuário esquecer o endereço IP do equipamento, este pode ser localizado no display de serviço do GP3320. As figuras a seguir mostram as ações necessárias para efetuar o procedimento. Caso o dispositivo utilizado não possua display entre em contato com a BCM.

Deslize o dedo sobre a área central do display



Selecione a aba "Rede" para exibir o IP

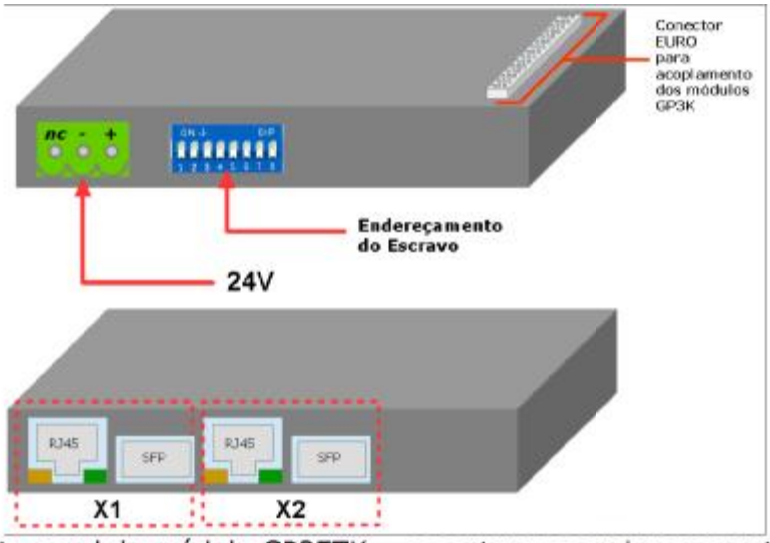
Além do endereço IP do controlador, o display de serviço mostra também a versão do Sistema Operacional instalado e a data/hora do CLP.

Para voltar à operação normal do IHM, pressionar um dos dois botões:

- **Reboot: Reinicia o CLP desde o S.Operacional, como se fosse ligada a alimentação**
- **Reset: Reinicia o programa do usuário**
- **Voltar: Volta ao IHM normal, sem alterar o funcionamento em curso**

3.3. O módulo EtherCAT

O módulo GP3ETK é um dispositivo EtherCat slave, usado para estender a capacidade de E/S dos controladores da série GP3320. A implementação do EtherCAT master é feita por software no próprio CLP. A rede EtherCat suporta a conexão de até 256 módulos GP3ETK, onde cada um permite a instalação local de até 4 módulos de E/S padrão da série GP3000.



Vista geral do módulo GP3ETK – conectores e endereçamento

3.3.1. Características técnicas

Portas Ethercat	**Duas portas Fast Ethernet, full-duplex 100 Mbit/s:** – **X1(IN):** – **- 100BASE-TX (até 100 m entre 2 nós) ou** – **- 100BASE-FX (até 10 km entre 2 nós)** – **X2(OUT):** – **- 100BASE-TX (até 100 m entre 2 nós) ou** – **- 100BASE-FX (até 10 km entre 2 nós)**
Alimentação	**24Vcc nominal (22 a 30V), 150mA (só módulo ETK), máximo de 1A com todos os módulos**
Temperatura de operação	-20 a +70°C
Dimensões	**140 x 92 x 40mm**
Peso	**350g**

3.3.2. Endereçamento do módulo GP3ETK

O endereçamento de cada módulo GP3ETK do sistema é feito através das chaves DIP. São aceitos endereços na faixa 0 a 255, codificados de forma binária nas chaves, onde a chave 1 é a LSB e a chave 8 é a MSB.
Observar que não deve ser nunca feito o endereçamento de dois módulos com o mesmo número e não há necessidade de que seja endereçados em ordem.

3.3.3. Considerações sobre a instalação da rede EtherCat

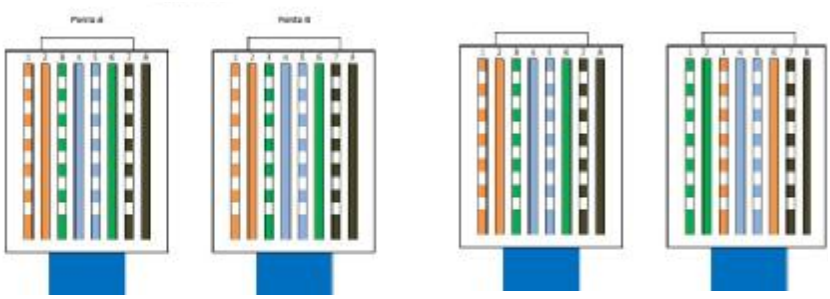
A seguir são apresentadas recomendações relacionadas à instalação do equipamento em rede EtherCAT. Detalhes sobre as características dos componentes utilizados para instalação podem ser obtidos juntamente ao ETG.

3.3.3.1. Cabo e pinagem

Características recomendadas para o cabo utilizado na instalação:

- **Cabo padrão Ethernet, 100Base-TX (FastEthernet), CAT 5e;**
- **Utilizar cabo blindado;**
- **Pode-se utilizar cabos com conexão direta ou crossover;**
- **Comprimento máximo para conexão entre equipamentos: 100 m.;**

Pino RJ-45	Sinal	Direção	Pino RJ-45
1	TD+	--->	1
2	TD-	--->	2
3	RD+	<---	3
4	--		4
5			5
6	RD-		6
7			7
8			8

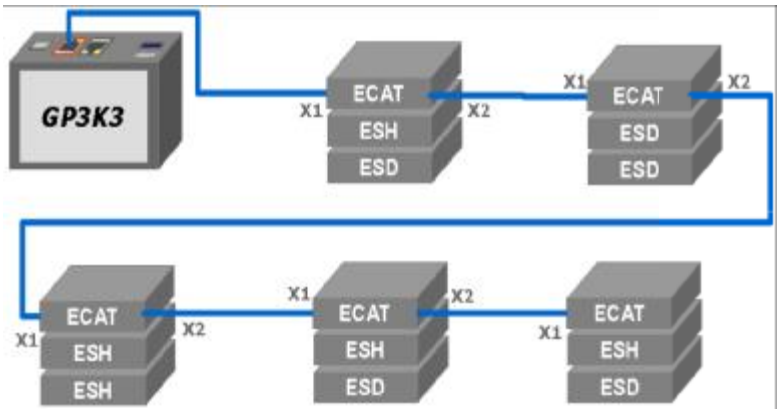


Cabo com conexão direta **Cabo crossover**

3.3.3.2. Topologia da rede

. Para a ligação do escravo EtherCAT, é necessário observar o conector RJ-45 para fazer a ligação:

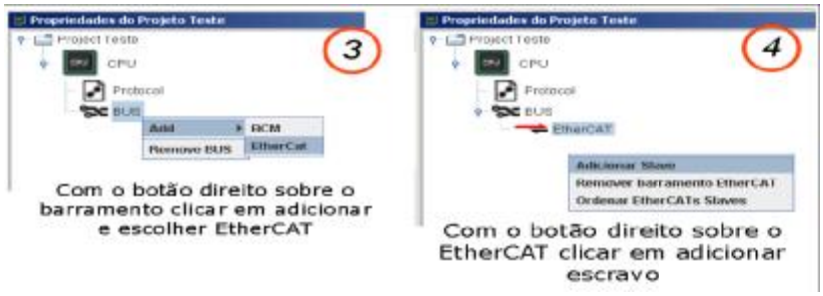
- **A rede sempre inicia pelo mestre EtherCAT.**
- **O conector X1 (IN) deve ser ligado sempre no segmento da rede que leva ao mestre EtherCAT.**
- **O conector X2 (OUT) deve ser ligado sempre no segmento da rede que leva aos demais escravos EtherCAT.**
- **Se houver ligação em anel para redundância, o conector X2 (OUT) do último escravo pode ser conectado à segunda porta do mestre EtherCAT, caso contrário deve ficar desconectado.**

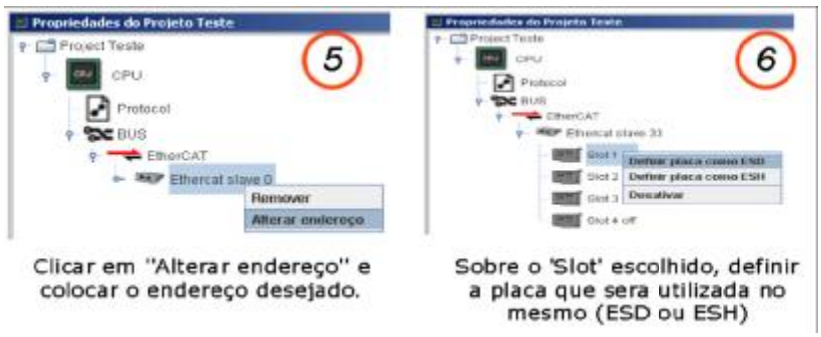


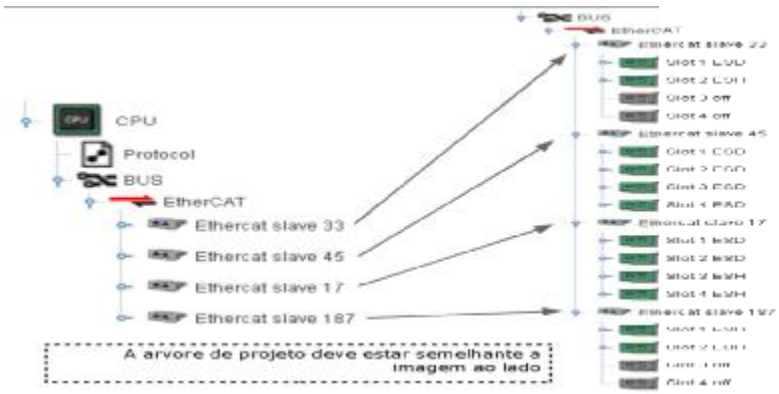
3.3.4. Configurando pontos de E/S EtherCat no Solver

Atenção: Ao configurar a rede Ethercat no GP3320, o canal Ethernet não poderá ser usado simultaneamente para outra finalidade (Modbus TCP, por exemplo). Além disso, a rede usada para ligar o GP3320 aos módulos GP3ETK deverá ser exclusiva – não pode ser usada a rede corporativa para essa finalidade!

Adicionando escravos EtherCAT no Solver





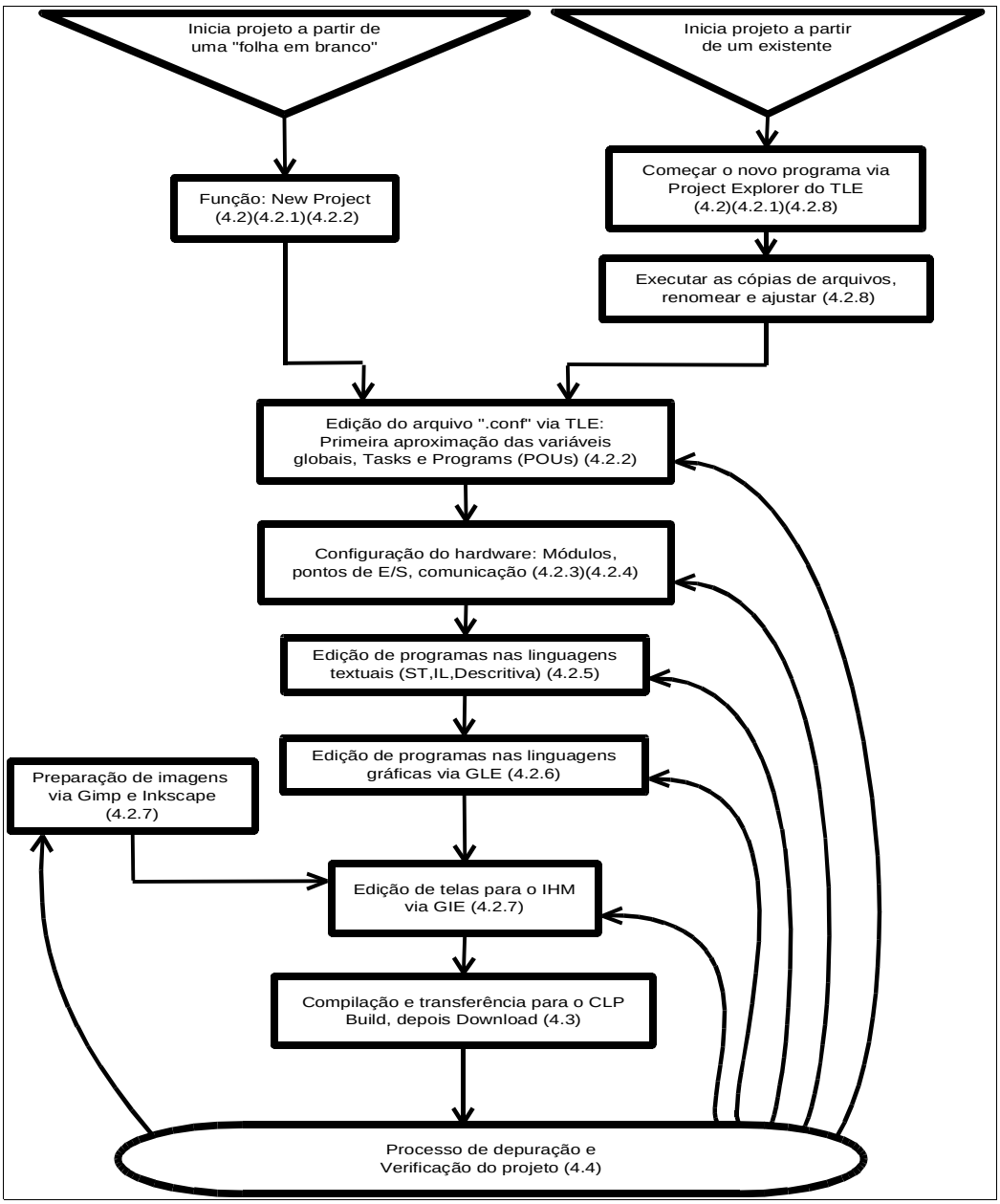


4. DESENVOLVIMENTO DE APLICAÇÕES E RECURSOS DE PROGRAMAÇÃO

Os equipamentos da linha GP3320 são programados através da ferramenta Solver, um avançado ambiente de desenvolvimento integrado (IDE) multi-plataforma (Windows/Linux) para aplicações de automação com suporte às linguagens previstas na norma IEC61131-3.

4.1. Fluxograma para desenvolvimento de aplicações

O diagrama a seguir mostra o fluxo recomendado para desenvolvimento de uma aplicação para o GP3320. Ao lado de cada bloco aparece o item do manual que descreve a operação em detalhes:



4.2. Aderência à norma IEC61131-3

Os controladores programáveis da série GP3320, em conjunto com o ambiente de desenvolvimento Solver, suportam as cinco linguagens padronizadas pela norma IEC61131-3 (IL, LD, SFC, FBD e ST), além da linguagem Descritiva BCM. Os quadros a seguir detalham os recursos, funções e tipos de dados suportados pelo GP3320 de uma forma alinhada aos itens da norma IEC61131-3.

Nro.	Número da tabela e título	Atendimento						Notas
	Descrição da característica	SFC	ST	BCM	IL	LD	FBD	
	Tabela 1 – Character set							
1	ISO/IEC 10646:2012, Information technology – Universal Coded Character Set (UCS)	x	x	x	x	x	x	
2a	Lower case characters a:, b, c...	x	x	x	x	x	x	Não suporta a observação “a”
2b	Number sign: # See table 5							
2c	Dollar sign: $ See table 5	x	x	x	x	x	x	
	Tabela 2 – Identifiers							
1	Upper case letters and numbers: IW215	x	x	x	x	x	x	
2	Upper case and lower case letters, numbers, embedded under-score	x	x	x	x	x	x	
3	Upper and lower case, numbers, leading or embedded under-score	x	x	x	x	x	x	
	Tabela 3 – Comments							
1	Single-line comment //...	x	x	x	x			
2a	Multi-line comment (* ... *)	x	x	x	x			
	Tabela 4 - Pragma							
1	Pragma with curly brackets { ... }	x	x	x	x	x		
	Tabela 5 – Numeric literals							
1	Integer literal: -1	x	x	x	x			
2	Real literal: -12.0	x	x	x	x			
3	Real literal with exponent: -1.34E-12	x	x	x	x			
4	Binary literal: 2#1111_1111	x	x	x	x			
6	Hexadecimal literal 16#FF	x	x	x	x			
7	Boolean zero and one	x	x	x	x	x	x	
8	Boolean FALSE and TRUE	x	x	x	x	x	x	
9	Typed literal: INT#-123 INT#167FFF WORD#167FFF WORD#16#AFF WORD#1234 UINT#16#89AF CHAR#16#41 BOOL#0 BOOL#1 BOOL#FALSE BOOL#TRUE	x	x	x	x			
	Tabela 8 – Duration literals	SFC	ST	BCM	IL	LD	FBD	
1a	d Day	x	x	x	x			
1b	h Hour	x	x	x	x			
1c	m Minute	x	x	x	x			

1d	s	Second	x	x	x	x			
1e	ms	Milisecond	x	x	x	x			
1f	us	Microsecond	x	x	x	x			
1g	ns	Nanosecond	x	x	x	x			
2a	Short prefix T#14ms T#-14ms LT#17.7s T#14.7h t#17.7d t#25h15m lt#5d14h12m18s8.5ms t#12h4m34ms230us400ns		x	x	x	x			
2b	Long prefix TIME#14ms TIME#-14ms time#14.7s		x	x	x	x			
3a	Short prefix t#25h_15m t#5d_14h_12m_18s_3.5ms LTIME#5m_30s_500ms_100.1us		x	x	x	x			Não suporta valores negativos; Não suporta prefixos LT;
3b	Long prefix TIME#25h_15m ltime#5d_14h_12m_18s_3.5ms		x	x	x	x			Não suporta valores negativos; Não suporta prefixos LTIME;
	Tabela 9 – Date and time of day literals								
1a	Date literal (long prefix) DATA#1984-06-25 date#2010-09-22		x	x	x	x			
1b	Data literal (short prefix) D#1984-06-25		x	x	x	x			
3a	Time of day literal (long prefix) TIME_OF_DAY#15:36:55.36		x	x	x	x			Não suporta frações de sgundos
3b	Time of day literal (short prefix) TOD#15:36:55.36		x	x	x	x			Não suporta frações de segundos
5a	Date and time literal (long prefix) DATE_AND_TIME#1984-06-25-15:36:55.360227400		x	x	x	x			Não suporta frações de segundos
5b	Date and time literal (short prefix) DT#1984-06-25-15:36:55.360_227_400		x	x	x	x			Não suporta frações de segundos
	Tabela 10 – Elementary data types								
1	Boolean	BOOL	x	x	x	x	x		
2	Short integer	SINT	x	x	x	x			
3	Interger	INT	x	x	x	x			
4	Double integer	DINT	x	x	x	x			
5	Long integer	LINT	x	x	x	x			
6	Unsigned short integer	USINT	x	x	x	x			
7	Unsigned integer	UINT	x	x	x	x			
8	Unsigned double interger	UDINT	x	x	x	x			
9	Unsigned long integer	ULINT	x	x	x	x			
10	Real numbers	REAL	x	x	x	x			
11	Long reals	LREAL	x	x	x	x			
12a	Duration	TIME	x	x	x	x			
13a	Date(only)	DATE	x	x	x	x			
14a	Time of day (only) TIME_OF_DAY or TOD		x	x	x	x			
15a	Date and time of day DATE_AND_TIME or DT		x	x	x	x			
18	Bit string of length 8	BYTE	x	x	x	x			
19	Bit string of length 16	WORD	x	x	x	x			
20	Bit string of lenght 32	DWORD	x	x	x	x			
21	Bit string of lenght 64	LWORD	x	x	x	x			

	Tabela 12 – Reference operations	SFC	ST	BCM	IL	LD	FBD	
1	Declaration of a reference type TYPE myRefType: REF_TO_INT; END_TYPE		x	x	x			
2a	Assigment reference to reference <reference>:= <reference> myRefType1:= myRefType2;		x	x	x			
2b	Assigment reference to parameter of function, functionblock and method myFB (a:=myTefS1); The types shall be equal		x	x	x			
3a	REF(<variable>) Provides of the typed reference to the variable myReffA1:= REF (A1);		x	x	x			
3b	REF(<function block instance>) Provides the typed reference to the function block or class instance myRefFB1:= REF(myFB1)		x	x	x			
4	<reference> Provides the content of the variable or the content of the instance to which the reference variable contains the reference. MyInt:= myA1Ref^[12];		x	x	x			
	Tabela 13– Declaration of variables							
1	Variable with elementary data type VAR MYBIT: BOOL; OKAY: STRING[10]; VALVE_POS AT %Q28: INT; END_VAR		x	x	x	x		Não suporta variável posicionada
3	Array VAR BITS: ARRAY[0..7] OF BOOL; TBT: ARRAY[1..2, 1..3] OF INT; OUTA AT %QW6: ARRAY[0..9] OF INT; END_VAR		x	x	x	x		
4	Reference VAR myRefInt: REF_TO_INT; END_VAR		x	x	x			
	Tabela 14 – Initialization of variables							
1	Initialization of a variable with elementary data type VAR MAYBIT: BOOL := 1; OKAY: STRING[10] := 'OK'; VALVE_POS AT %QW28: INT:= 100; END_VAR		x	x	x	x		
4	Declaration and initialization of constants VAR CONSTANT PI: REAL:=3.141592; PI2 REAL := 2.0*PI; END_VAR		x	x		x		Não suporta expressões
6	Initialization of a reference		x	x	x			

	VAR myRefInt: REF_TO_INT := REF(myINT); END_VAR							
	Tabela 19 – Function declaration	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1a	Without result FUNCTION ..... END_FUNCTION		x	x	x			
2b	With result FUNCTION <name> : <data type> ..... END_FUNCTION		x	x	x			A variável de retorno deve ser explicitamente declarada no interior da função
2a	Inputs VAR_INPUTS .... END_VAR		x	x	x	x		
2b	Outputs VAR_OUTPUT .... END_VAR		x	x	x	x		
2c	In-outs VAR_IN_OUT .... END_VAR		x	x	x	x		
2e	Temporary variables VAR .... END_VAR		x	x	x			
2f	External variables VAR_EXTERNAL .... END_VAR		x	x	x			
2g	External constants VAR_EXTERNAL CONSTANT .... END_VAR		x	x	x			
3a	Initialization of inputs		x	x	x	x		
3b	Initialization of outputs		x	x	x	x		
3c	Initialization of temporary variables		x	x	x	x		
	Tabela 20 – Function call							
1a	Complete formal call (textual only) A:= LIMIT (EN:= COND, IN:= B, MN:=0, MX:=5, EN0 => TEMPL);		x	x	x			
1b	Incomplete formal call (textual only) A:= LIMIT (IN:=B, MX:=5);		x	x	x			
2	Non-formal (textual only, fix order and complete) A:=LIMIT (B, 0, 5);		x	x	x			
3	Function without function result FUNCTION myFun // no type declaration VAR_INPUT x: INT; END_VAR; VAR_OUTPUT y: REAL; END_VAR; myFun(150, var); // Call		x	x	x			
	Tabela 21 – Typed and overloaded functions	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	

1a	Overloaded function ADD (ANY_Num to ANY_Num)		x	x	x			
1b	Conversion of inputs ANY_ELEMENT TO_INT		x	x	V			
2b	Conversion WORD_TO_INT		x	x	x			
	Tabela 22 – Data type conversion function							
1b	Overloaded conversion TO_output A:= TO_REAL(b);		x	x	x			
2c	Overloaded truncation TRUNC_output A:= TRUNC_INT(B);		x	x	x			
3b	Overloaded BCD_TO_output A:= BCD_TO_INT(B);		x	x	x			
4b	Overloaded TO_BCD_out A:= TO_BCD_WORD(B);		x	x	x			
	Tabela 28 – Numerical and arithmetic functions	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	ABS(x)		x	x	x			
2	SQRT(x)		x	x	x			
3	LN(x)		x	x	x			
4	LOG(x)		x	x	x			
5	EXP(x)		x	x	x			
6	SIN(x)		x	x	x			
7	COS(x)		x	x	x			
8	TAN(x)		x	x	x			
9	ASIN(x)		x	x	x			
10	ACOS(x)		x	x	x			
11	ATAN(x)		x	x	x			
12	ATAN2(x, y)		x	x	x			
	Tabela 29 –Arithmetic functions	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	Addition ADD, +		x	x	x			
2	Multiplication MUL, *		x	x	x			
3	Subtraction SUB, -		x	x	x			
4	Division DIV, /		x	x	x			
5	Módulo MOD		x	x	x			
6	Exponentiation EXPT, **		x	x	x			
7	Move :=		x	x	x			
	Tabela 30 – Bit shift functions	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	Shift left SHL		x	x	x			
2	Shift right SHR		x	x	x			
3	Rotation left ROL		x	x	x			
4	Rotation right ROR		x	x	x			
	Tabela 31 – Bitwise Boolean functions	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	And AND,&		x	x	x			
2	Or OR		x	x	x			
3	Exclusive Or XOR		x	x	x			
4	Not NOT		x	x	x			

	Tabela 32 – Selection functions	SFC	ST	BCM	IL	LD	FBD	
2	Binary selection SEL		x	x	x			
3	Extensible maxmum function MAX		x	x	x			
4	Extensible minimum function MIN		x	x	x			
5	Limiterx LIMIT		x	x	x			
6	Extensible multiplexer MUX		x	x	x			
	Tabela 33 – Comparasion functions	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	Decreasing sequence GT		x	x	x			
2	Monotonic sequence GE		x	x	x			
3	Equality EQ		x	x	x			
4	Monotonic sequence LE		x	x	x			
5	Increasing LT		x	x	x			
6	Inequality NE		x	x	x			
	Tabela 37 – Functions for endianess conversion	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	TO_BIG_ENDIAN A:= TO_BIG_ENDIAN(B);		x	x	x			
2	TO_LITTLE_ENDIAN B:= TO_LITTLE_ENDIAN(A);		x	x	x			
3	BIG_ENDIAN_TO A:= FROM_BIG_ENDIAN(B);		x	x	x			
4	LITTLE_ENDIAN_TO A:= FROM_LITTLE_ENDIAN(B);		x	x	x			
	Tabela 39 – Validate functions	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	IS_VALID VAR R: REAL; END_VAR IF IS_VALID® THEN ...		x	x	x			
2	IS_VALID_BCD VAR W: WORD; END_VAR IF IS_VALID_BCD(W) THEN ...		x	x	x			
	Tabela 40 – Function block type declaration							
1	Declaration of function block type FUNCTION_BLOCK ... END_FUNCTION_BLOCK		x	x	x			
2a	Declaration of inputs VAR_INPUT ... END_VAR		x	x	x			
2b	Declaration of outputs VAR_OUTPUT ... END_VAR		x	x	x			
2c	Declaration of in-outs VAR_IN_OUT ... END_VAR		x	x	x			
2e	Declaration of static variables VAR ... END_VAR		x	x	x			
2f	Declaration of external variables VAR_EXTERNAL ... END_VAR		x	x	x			
2g	Declaration of external variables VAR_EXTERNAL CONSTANT ... END_VAR		x	x	x			
3a	Initialization of inputs		x	x	x			

	VAR_INPUT MN: INT:= 0;							
3b	Initialization of outputs VAR_OUTPUT RES: INT:=1;		x	x	x			
3c	Initialization of static variables VAR B: REAL:= 12.1;		x	x	x			
4a	Declaration of RETAIN qualifier on input variables VAR_INPUT RETAIN X: REAL; END_VAR		x	x	x			
4b	Declaration of RETAIN qualifier on output variables VAR_OUTPUT RETAIN X: REAL; END_VAR		x	x	x			
4c	Declaration of NON_RETAIN qualifier on input variables VAR_INPUT NON_RETAIN X: REAL; END_VAR		x	x	x			
4d	Declaration of NON_RETAIN qualifier on output variables VAR_OUTPUT NON_RETAIN X: REAL; END_VAR		x	x	x			
4e	Declaration of RETAIN qualifier on static variables VAR RETAIN X: REAL; END_VAR		x	x	x			
4f	Declaration of NON_RETAIN qualifier on static variables VAR NON_RETAIN X: REAL; END_VAR		x	x	x			
5a	Declaration of RETAIN qualifier on local FB instances VAR RETAIN TMR1: TON; END_VAR		x	x	x			
5b	Declaration of RETAIN qualifier on local FB instances VAR NON_RETAIN TMR1: TON; END_VAR		x	x	x			
	Tabela 41 – Function block instance declaration	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	Declaration of FB instance VAR FB_instance_1, FB_instance_2: my FB_Type; T1, T2, T3: TON; END_VAR		x	x	x			
	Tabela 42 – Function block call							
1	Complete formal call (textual only) YourCTU(EN:= not B, CU:= r, PV:= c1, ENO=> next, Q=> out, CV => c2);		x	x	x			
2	Incomplete formal call (textual only) YourCTU (Q => out,CV => c2);		x	x	x			
6a	Textual call with separate assigment of input FB_Instance.Input := x;		x	x	x			
7	Textual output read after FB call		x	x	x			
8a	Textual output assigned in FB call		x	x	x			

	Tabela 43 – Standard bistable function blocks							
1a	**Bistable function block (set dominant) SR (S1, R, Q1)**		**x**	**x**	**x**			
1b	**Bistable function block (set dominant) with long input names SR(SET1, RESET, Q1)**		**x**	**x**	**x**			
2a	**Bistable function blocks (reset dominant) RS (S, R1, Q1)**		**x**	**x**	**x**			
2b	**Bistable function blocks (reset dominant) with long input names RS (SET, RESET1, Q1)**		**x**	**x**	**x**			
	Tabela 44 – Standard edge detection function blocks							
1	**Rising edge detector R_TRIG (CLK, Q)**		**x**	**x**	**x**			
2	**Falling edge detector F_TRIG(CLK, Q)**		**x**	**x**	**x**			
	Tabela 45 – Counters							
1a	**CTU_INT (CU, R, PV, Q, CV) or CTU**		**x**	**x**	**x**			
1b	**CTU_DINT PV, CV: DINT**		**x**	**x**	**x**			
1c	**CTU_LINT PV, CV: LINT**		**x**	**x**	**x**			
1d	**CTU_DINT PV, CV: UDINT**		**x**	**x**	**x**			
1e	**CTU_ULINT PV, CV: ULINT**		**x**	**x**	**x**			
2a	**CTD_INT(CD, LD, PV, Q, CV) or CTD**		**x**	**x**	**x**			
2b	**CTD_DINT PV, CV: DINT**		**x**	**x**	**x**			
2c	**CTD_LINT PV, CV: LINT**		**x**	**x**	**x**			
2d	**CTD_UINT PV, CV: UDINT**		**x**	**x**	**x**			
2e	**CTD_ULINT PV, CV: UDINT**		**x**	**x**	**x**			
3a	**CTUD_INT(CD, LD, PV, Q, CV) or CTUD**		**x**	**x**	**x**			
3b	**CTUD_DINT PV, CV: DINT**		**x**	**x**	**x**			
3c	**CTUD_LINT PV, CV: LINT**		**x**	**x**	**x**			
3d	**CTUD_UDINT PV, CV: UDINT**		**x**	**x**	**x**			
3e	**CTUD_ULINT PV, CV: ULINT**		**x**	**x**	**x**			
	Tabela 46 – Standard timer function blocks	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1a	**Pulse, overloaded TP**		**x**	**x**	**x**			
2a	**On-delay, overloaded TON**		**x**	**x**	**x**			
3a	**Off-delay, overloaded TOF**		**x**	**x**	**x**			
	Tabela 47 – Program declaration							
1	**Declaration of a program PROGRAM ... END PROGRAM**		**x**	**x**	**x**	**x**		
2a	**Declaration of input VAR_INPUT ... AND_VAR**		**x**	**x**	**x**	**x**		
2b	**Declaration of outputs VAR_OUTPUT .. AND_VAR**		**x**	**x**	**x**	**x**		
2c	**Declaration of in-outs VAR_IN_OUT ... END_VARS**		**x**	**x**	**x**	**x**		
2e	**Declaration of static variables VAR ... END_VAR**		**x**	**x**	**x**	**x**		
2f	**Declaration of external variables VAR_EXTERNAL ... END_VAR**		**x**	**x**	**x**	**x**		
3a	**Initialization of inputs**		**x**	**x**	**x**	**x**		

3b	**Initialization of outputs**		**x**	**x**	**x**	**x**		
3c	**Initialization of static variables**		**x**	**x**	**x**	**x**		
4a	**Declaration of RETAIN qualifier on input variables**		**x**	**x**	**x**	**x**		
4b	**Declaration of RETAIN qualifier on output variables**		**x**	**x**	**x**	**x**		
4c	**Declaration of NON_RETAIN qualifier on input variables**		**x**	**x**	**x**	**x**		
4d	**Declaration of NON_RETAIN qualifier on output variables**		**x**	**x**	**x**	**x**		
4e	**Declaration of RETAIN qualifier on static variables**		**x**	**x**	**x**	**x**		
4f	**Declaration of NON_RETAIN qualifier on static variables**		**x**	**x**	**x**	**x**		
5a	**Declaration of RETAIN qualifier on local FB instances**		**x**	**x**	**x**			
5b	**Declaration of NON_RETAIN qualifier on local FB instances**		**x**	**x**	**x**			
	Tabela 54 – SFC step							
1a	**Step – graphical form with directed links**							
1b	**Intial step – graphical from with directed link**							
2a	**Step – textual from without directed links**	**x**						
2b	**Initial step – textual form without directed links** **INITIAL_STEP ***:** **(* Step body *)** **END_STEP**	**x**						
3a	**Step flag – geretal form:** *****.X = BOOL #1 when *** is active, BOOL #0 otherwise**	**x**						
3b	**Step flag direct connection of BOLLEAN variable ***.X to right side of step**	**x**						
4	**Step elapsed time – general form:** *****.T a variable of type TIME**	**x**						
	Tabela 55 – SFC step	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1a	**Transition condition physically or logically adjacent to the transition using ST laguage**	**x**						
2a	**Transition condition physically adjacent to the transition using LD language**							
3a	**Transition condition physically or logically adjacent to the transition using FBD language**							
4a	**Use of connector**							
5a	**Transition condition: Using LD laguage**							
6a	**Transition condition: Using FBD language**							
7b	**Textual equivalent of feature 1 using ST language**	**x**						
8b	**Textual equivalent of feature 1 using IL language**							
9a	**Use of transition name**	**x**						
10a	**Transition condition using LD language**							
11a	**Transition condition using FBD language**							
12b	**Transition condition using IL language**							
13b	**Transition condition using ST language**	**x**						
	Tabela 56 – SFC declaration of actions							

1	Any Booolean variable declared in a Var or VAR_OUTPUT block, or their graphical equivalents, can be an action		x						
2l	Graphical declaration in LD language								
2s	Inclusion os SFC eelements in action		x						
2f	Graphical declaration in FBD language								
3s	Textual declaration in ST language		x						
3i	Testual declaration in IL language								
	Tabela 57 – Step/Action association								
1	Action block physically or logically adjacent to the step								
2	Concatanated action blocks physically or logically adjacent to the step								
3	Textual step body		x						
4a	Action block “d” field		x						
	Tabela 58 – Action block								
1a	“a”: Qualifier as per 6.7.4.5		x						
2	“b”: Action name		x						
3b	“c”: Boolean “indicator” variables (deprecated)		x						
4i	IL language		x						
4s	ST language		x						
4l	LD language								
4f	FBD language								
5l	Use of action blocks LD								
5f	Use of action blocks in FBD								
	Tabela 59 – Action qualifiers								
1	Non-stored (null qualifier)	None	x						
2	Non-stored	N	x						
3	Overriding Reset	R	x						
4	Set(Stored)	S	x						
5	time Limited	L	x						
6	time Delayed	D	x						
7	Pulse	P	x						
8	Stored and time Delayed	SD	x						
9	Delayed and Stored	DS	x						
10	Stored and time Limited	SL	x						
11	Pulse (rising edge)	P1	x						
12	Pulse (falling edge)	P0	x						
	Tabela 60 – Action control features		SFC	ST	BCM	IL	LD	FBD	
1	With final scan per figure 22a and 23a		x						
2	Without final scan per figure 22b and 23b		x						
	Tabela 61 – Sequence evolution – graphical								
1	Single sequence								
2a	Divergence of sequence with left to right priority								
2b	Divergence of sequence with numbered branches								

2c	Divergence of sequence with mutual exclusion								
3	Convergence of sequence								
4a	Simultaneous divergence after a single transition								
4b	Simultaneous divergence after conversion								
4c	Simultaneous convergence before one transition								
4d	Simultaneous convergence before a sequence selection								
5a, b, c	Sequence skip								
6a, b, c	Sequence loop								
7	Directional arrow								
	Tabela 62 – Configuration and resource declaration								
1	CONFIGURATION ... END_CONFIGURATION	x	x	x	x	x			
2	VAR_GLOBAL ... END_VAR within CONFIGURATION	x	x	x	x	x			
5a	Periodic TASK	x	x	x	x	x			
5b	Non-periodic TASK	x	x	x	x	x			
6a	WITH for PROGRAM to TASK association	x	x	x	x	x			
6c	PROGRAM with no TASK association	x	x	x	x	x			
8a	Connection of directly represented variables to PROGRAM inputs	x	x	x	x	x			
8b	Connection of GLOBAL variables to PROGRAM inputs	x	x	x	x	x			
9a	Connection of PROGRAM outputs to directly represented variables	x	x	x	x	x			
9b	Connection of PROGRAM outputs to GLOBAL variables	x	x	x	x	x			
	Tabela 63 – Task	SFC	ST	BCM	IL	LD	FBD		
1a	Textual declaration of periodic TASK	x	x	x	x	x			
1b	Texual declaration of non-periodic TASK	x	x	x	x	x			
3a	Textual association with PROGRAMS	x	x	x	x	x			
5b	Preemtive schedulling	x	x	x	x	x			
	Tabela 67 – Parenthesized expression of IL language								
1	Parenthesized expression beginning with explicit operator				x				
2	Parenthesized expression (short from)				x				
	Tabela 68 – Instruction list operators								
	Operator	Modifier	Explanation						
1	LD	N	Set current result equal to operand				x		
2	ST	N	Store the current result to operand location				x		
3	S, R		Set operand to 1 if current result is boolean 1 Reset operand to 0 if current result is boolean 1				x		

4	AND	N, (	Logical AND				x			
5	&	N, (	Logical AND				x			
6	OR	N, (	Logical OR				x			
7	XOR	N, (	Logical exclusive OR				x			
8	NOT		Logical negation				x			
9	ADD	(	Addition				x			
10	SUB	(	Subtraction				x			
11	MUL	(	Multiplication				x			
12	DIV	(	Division				x			
13	MOD	(	Modulo-division				x			
14	GT	(	Comparison: >				x			
15	GE	(	Comparison: >=				x			
16	EQ	(	Comparison: =				x			
17	NE	(	Comparison: <>				x			
18	LE	(	Comparison: <=				x			
19	LT	(	Comparison: <				x			
20	JMP	C, N	Jump to a label				x			
21	CAL	C, N	Call function block				x			
22	RET	C, N	Retrun from called function, function block or program				x			
23	)		Evaluate deferred operation				x			
24	ST?		Assigment attempt Store with test				x			

	Tabela 69 – Calls for IL language	SFC	ST	BCM	IL	LD	FBD	
1a	Function block call with non-formal parameter CALL C10 (%IX10, FALSE, A, OUT, B) CALL CMD_TMR(%IX5, T#300ms, OUT, ELAPSED)				x			
1b	Function block call with formal parameter list CAL C10 (CU:=%IX10, R:=FALSE, PV:=A, Q=>B) CAL CMD_CMD_TMR (IN: =%IX5, PT:=T#300ms, Q:=>OUT, ET=>ELAPSED, EBO=>ERR)				x			
2	Function block call with load/store of standard input parameter LD A ADD 5 ST C10.PV LD %IX10 ST C10.CU CAL C10 LD C10.CV				x			
3a	Function call with formal parameter list				x			

	LIMIT (EN:= COND, IN:= B, MN:= 1, MX:= 5, ENO=> TEMPL) ST A										
3b	Function call with non-formal parameter list LD 1 LIMIT B,5 ST A							x			
4a	Method call with formal parameter list FB_INST.M1 (E:= COND, IN:= B, MN:= 1, MX:= 5, ENO=> TEMPL) ST A							x			
	Tabela 70 – Standard function block operators for IL language										
	Function Block	Input Operator	Output Operator								
1	SR	S1, R	Q					x			
2	RS	S, R1	Q					x			
3	F/R_TRIG	CLK	Q					x			
4	CTU	CU, R, PV	CV, Q, also RESET					x			
5	CTD	CD, PV	CV, Q					x			
6	CTUD	CU, CD, R, PV	CV, QU, QD, also RESET					x			
7	TP	IN, PT	CV, Q					x			
8	TON, IN, PT	IN, PT	CV, Q					x			
9	TOF	IN, PT	CV, Q					x			
	Tabela 71 – Operators of the ST language				SFC	ST	BCM	IL	LD	FBD	
	Description Operator	Symbol	Example	Precedence							
1	Parentheses	(expression)	(A+B/C), (A+B)/C, A/(B+C)	11 (Highest)		x	x				
2	Evaluation fo a result of function and method – if a result is declared	Identifier (parameter list)	LN(A), MAX(X,Y), myclass.mymethod(x)	10		x	x				
3	Dereference	^	R^	9		x	x				
4	Negation	-	-A, - A	8		x	x				
5	Unary Plus	+	+B, + B	8		x	x				
6	Complement	NOT	NOT C	8		x	x				
7	Exponentiation	**	A**B, B ** B	7		x	x				
8	Multiply	*	A*B, A * B	6		x	x				
9	Divide	/	A/B, A / B /D	6		x	x				

10	Modulo	MOD	A MOD B	6		x	x				
11	Add	+	A+B, A+ B + C	5		x	x				
12	Subtratct	-	A-B, A – B – C	5		x	x				
13	Comparasion	<, >, <=, >=	A<B	4		x	x				
14	Equality	=	A=B, A+B & B=C	4		x	x				
15	Inequality	<>	A<>B, A <> B			x	x				
16a	Boolean AND	&	A&B, A & B, A& B & C	3		x	x				
16b	Boolean AND	AND	A AND B	3		x	x				
17	Boolean Exclusive OR	XOR	A XOR B	2		x	x				
18	Boolean OR	OR	A OR B			x	x				
	Tabela 72 – ST language statements				SFC	ST	BCM	IL	LD	FBD	
1	Variable : = expression					x					
1a	Variable and expression of elementary data type A:=B; CV:=CV+1; C:= SIN(X);					x					
1b	Variables and expression of different elementary data types with impllicit type conversion according Figure 11 A_REAL := B_Int					x					
1d	Instances of function block type A_Instance := B_Instance;					x					
2a	Function call FCT(17);					x					
2b	Function block call and FB output CMD_TMR(IN:=bIn1, PT:= T#300ms) A:= CMD_TMR.Q;					x					
3	RETURN					x					
4	IF ... THEN ... ELSIF ... THEN ... ELSE ... END_IF D:= B*B – 4.0*A*C; IF D < 0.0 THEN NROOTS := 0; ELSEIF D = 0.0 THEN NROOTS := 1; X1 := - B/(2.0*A); ELSE NROOTS := 2; X1 := (-B+SQRT(D))/(2.0)*A); X2 := (-B-SQRT(D))/(2.0)*A); END_IF;					x					
5	CASE ... OF ... ELSE ... END_CASE TW:= WORDBCD_TO_INT(TUMBWHEEL); TW_ERROR := 0; CASE TW OF 1,5:DISPLAY := OVEN_TEMP; 2:DISPLAY := MOTOR_SPEED; 3:DISPLAY:= GROSS – TARE;					x					

	4,6, .. 10: DISPLAY:= STATUS(TW-4); ELSE DISPLAY := 0; TW_ERROR:=1; END_CASE; QW100 : INT_TO_BCD(DISPLAY);							

6	FOR ... TO ... BY ... DO ...
	...
	END_FOR
	
	J:=101;
	FOR I:=1 TO 100 BY 2 DO
	IF WORDS[I] = 'KEY' THEN
	J:=I;
	EXIT;
	END_IF;
	END_FOR;
			x					

7	WHILE ... DO
	...
	END_WHILE
	
	J:=1;
	WHILE J <= 100 & WORD[J] <> 'KEY' DO
	J:=J+2;
	END_WHILE;
			x					

8	REPEAT ... UNTIL ... END_REPEAT J:=-1; REPEAT J:= J+2; UNTIL J= 101 OR WORD[J] = 'KEY' END_REPEAT;		x					
9	CONTINUE J:=1; WHILE (J<= 100 AND WORD[J] <> 'KEY') DO .. IF (J MOD 3 = 0) THEN CONTINUE END_IF; ... END_WHILE;		x					
10	EXIT an interation		x					
11	Empty Statement ;		x					
	Tabela 73 – Graphic execution control elements	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1a	Unconditional jump - FBD language							
1b	Unconditional jump - LD language							
2a	Conditional jump - FBD language							
2b	Conditional jump - LD language							
3a	Conditional return - LD language							
3b	Conditional return – FBD language							
4	Unconditional return – LD language							
	Tabela 74 – Power rails and link elements	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	Left power rail (with attached horizontal link)					x		
2	Right power rail (with attached horizontal link)					x		

3	**Vertical link (with attached horizonatl link)**					**x**		
	Tabela 75 – Contacts	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	**Normally open contact**					**x**		
2	**Nomally closed contact**					**x**		
3	**Positive transition-sensing contact**					**x**		
4	**Negative transition-sensing contact**					**x**		
5a	**Compare contact, (typed)**					**x**		
5b	**Compare contact, (overloaded)**					**x**		
	Tabela 76 – Coils	**SFC**	**ST**	**BCM**	**IL**	**LD**	**FBD**	
1	**Coil**					**x**		
2	**Negative coil**					**x**		
3	**Set (latch) coil**					**x**		
4	**Reset (unlatch) coil**					**x**		
8	**Positive transistion-sensing coil**					**x**		
9	**Negative transition-sensing coil**					**x**		

4.3. Gerenciamento de tarefas, tempos de execução e prioridades

O controlador GP3320 permite que o usuário configure múltiplas tarefas para agendar e priorizar a execução de cada elemento da lógica do programa. A fim de aproveitar da melhor forma a capacidade de processamento da CPU, cada parte do programa pode ser agendada para execução na periodicidade justa que o processo que está sendo automatizado exige.

No manual do Solver está descrito como são programadas as tarefas (tasks) e como estas são vinculadas às unidades lógicas (POUs) do programa do usuário. O objetivo aqui é descrever como o sistema operacional trata a execução de todo o conjunto de processos e orientar como o usuário pode fazer a configuração de forma otimizada para cada aplicação.

4.3.1 Como o GP3320 executa os programas

A capacidade de processamento da CPU é compartilhada em uma série de processos, cada um deles executado ciclicamente de forma independente:

- **Execução do programa do usuário configurado através das tasks {*1}**
- **Atendimento do IHM (display & touchscreen)**
- **Operações ativas de comunicação (configuradas pelo usuário) {*1}**
- **Atendimento dos pontos de E/S locais {*2}**
- **Atendimento dos pontos de E/S remotos (Ethercat) {*2}**
- **Overhead do sistema operacional**

{*1} As tarefas que compõem esses processos tem a sua periodicidade configurada pelo usuário. O sistema operacional distribui a capacidade da CPU de forma a cumprir os agendamentos do usuário e usa o tempo restante para os demais processos.

{*2} Os tempos de execução (ciclos) dos atendimentos de pontos de E/S dependem do número e tipo de módulos de E/S instalados.

4.3.2. Configuração do período de varredura de E/S

A programação do tempo de varredura dos barramentos é feita pela instrução BUSRATE, com a seguinte sintaxe:

BUSRATE(bus, tempo)

Onde:
***bus* - número do barramento (0:BCM, 1:Ethercat);**
***tempo* - é o valor mínimo entre duas varreduras do barramento em nano segundos;**

A cada passagem por esta instrução, o período é reconfigurado com o valor dos parâmetros. Se a instrução não for utilizada, a varredura do barramento fica sendo executada continuamente, na máxima velocidade possível.

4.3.3. Recomendações para otimização do programa do usuário

a) Com base na especificação do projeto e no conhecimento dos processos, dividir esses processos em diferentes POUs, organizando-os numa distribuição funcional. Isto facilita a programação, manutenção e planejamento do uso da CPU para atendimento de todas as tarefas.

b) Criar e configurar Tasks para conter os POUs previstos – configurar o período de execução (INTERVAL) de cada uma de forma coerente com as demandas de tempo do sistema, distribuindo os POUs entre as diferentes Tasks de acordo.

c) Quando houver uma demanda de alta velocidade, tipicamente associada à resposta a uma variação de entrada digital, considerar o uso de uma Task ativada por evento ao invés de ativada por um período fixo.

d) Durante a depuração do programa, usar os recursos descritos no item 4.2.2 deste manual para monitorar os tempos máximos de execução e condições de overrun. Essa avaliação irá mostrar as situações em que eventualmente os períodos das Tasks devam ser ajustados de forma a atender o tempo de resposta desejado.

e) Em processos críticos, pode ser importante incluir uma lógica específica de watchdog, que monitore em tempo real os tempos de execução, gerando registros e/ou alarmes que sinalizem a ultrapassgem dos tempos previstos para os processos.

4.4. Funções específicas para o GP3320

4.4.1. Temporizadores

O Solver fornece instruções de temporização, as quais correspondem a um tipo de relógio especializado usado para controlar uma sequência de eventos ou processos. Para utilizar os temporizadores disponíveis no Solver é necessário primeiramente criar uma instância do bloco de função *TIMER*. Podemos observar no exemplo abaixo como deve ser declarada:

```
VAR
  TIMER_INST : TIMER;
END VAR
```

Com a instância já declarada, neste caso com o nome de *TIMER_INST*, podemos fazer uso dos recursos fornecidos por esta instância, conforme descrito a seguir:

```
TIMER_INST.PV=[preset value];
```

Configura o contador com o valor pré-determinado ao ser carregado. O valor deve ser uma constante ou variável do tipo TIME. Exemplo: *TIMER_INST.PV:=T#1ms;*

```
TIMER_INST(^R:=1);
```

Reinicia a contagem de tempo, resetando o timer.

```
TIMER_INST(^LD:=1);
```

Realiza a recarga do contador para o valor pré-estabelecido em [PV].

```
TIMER_INST();
```

Atualiza o valor da instância com o valor corrente do temporizador.

```
TIMER_INST.CV=[test value];
```

Disponibiliza o valor atual do temporizador, para ser usado num teste IF ou como argumento numa operação. O valor de retorno será do tipo TIME.
Para fazer uso efetivo do valor corrente do temporizador é necessário primeiramente atualizar o valor da instância com o comando *TIMER_INST()*. Exemplo:

```
TIMER_INST();
IF TIMER_INST.CV>=T#20s THEN
        ..............
END_IF;
```

Observar que [test value] deve ser uma constante ou variável do tipo TIME!

4.4.2. Leitura do relógio/calendário

O Solver fornece instruções para leitura do relógio/calendário nas lógicas de automação. Para utilizar esta função específica do GP3320, é necessário primeiramente realizar a instanciação de uma variável do tipo RTC. Podemos observar no exemplo abaixo como deve ser declarada:

```
VAR
  RELOGIO : RTC;
END VAR
```

Com a variável já declarada, neste caso com o nome de *RELOGIO*, podem ser adquiridos os diversos valores relativos ao relógio/calendário:

```
RELOGIO();                  //Atualiza as variáveis de relógio na instância declarada.
DIA := RELOGIO.MDAY;
MES := RELOGIO.MON;
ANO := RELOGIO.YEAR;
HORA := RELOGIO.HOUR;
MINUTO := RELOGIO.^MIN;
SEGUNDO := RELOGIO.SEC;
MILISEGUNDO := RELOGIO.MILISEC;
```

As variáveis nas quais os valores são carregados tem nomes livres e devem ser declarados como variáveis do tipo UINT.

4.4.3. Uso de variáveis retentivas

O GP3320 oferece um espaço de 64kBytes de memória retentiva. O número de variáveis que pode ser armazenado depende do tipo de variável usada.
A configuração deve ser feita usando a diretiva *VAR RETAIN* no POU onde as variáveis são usadas e, caso sejam globais, também no arquivo *.conf.* No caso de variáveis globais, estas são declaradas nos POUs como *EXTERNAL* normalmente.
O exemplo a seguir mostra a declaração de variáveis retentivas globais e locais:

```
VAR_GLOBAL RETAIN           //Declaração no arquivo .conf.
   fibonacci : ULINT;
END_VAR


VAR RETAIN                  //Declaração no POU.
  Contador : UINT;
  ContadorLocal : UINT;
END_VAR
```

4.4.4. Cartão de memória

. **As funções do sistema de arquivos oferecem suporte para o sistema de arquivos padrão Windows FAT32. Com Isso, o Solver é capaz de fornece aos aplicativos a oportunidade de ter uma capacidade de armazenamento de até 256 arquivos. Isto possibilita o registro de parâmetros, dados, anotações de eventos, configurações de usuários, etc.**

ATENÇÃO!

O cartão de memória não deve ser retirado do GP3320 com o controlador ligado - Sempre desligue a alimentação antes de retirá-lo!

Funções disponíveis para a manipulação de arquivos em cartão de memória:

```
FILE_OPEN(NAME:=[NAME],ID=>[ID], ERROR=>[ERROR]);
```

- **[NAME] Constante ou variável tipo USINT com valor como nome do arquivo;**
- **[ID] Ponteiro devolvido para manipular as funções do cartão (variável tipo INT);**
- **[ERROR] Retorna 1 se houver falha na abertura (variável tipo BOOL);**

Abre um arquivo com um determinado ([NOME]) e devolve um ponteiro ([ID]) para que se possa executar funções de escrita e leitura no arquivo aberto. Se o arquivo com o nome determinado já existir, a função simplesmente devolve o ponteiro para o arquivo, caso contrário, o arquivo é criado.

```
FILE_READ(ID:=[ID], DATA=>[DATA]);
```

- **[ID] Ponteiro produzido na instrução de OPEN;**
- **[DATA] Dado retornado ao realizar uma leitura do cartão;**

Após a instrução de OPEN a instrução de leitura realiza a aquisição dos dados a partir da primeira linha do arquivo, e incrementa automaticamente uma linha após cada leitura;

```
FILE_WRITE(ID:=[ID], DATA:=[DATA]);
```

- **[ID] Ponteiro produzido na instrução de OPEN;**
- **[DATA] Dado para ser escrito no cartao;**

Após a instrução de OPEN a instrução de escrita realiza a gravação dos dados a partir da primeira linha do arquivo. Sucessivas operações irão escrever cada novo valor ao lado dos já existentes, separados por um espaço.

```
FILE_WRITE(ID:=[ID], NEW_LINE);
```

- **[ID] Ponteiro produzido na instrução de OPEN;**

Lança uma nova linha para registro dos próximos valores no arquivo indicado.

```
FILE_CLOSE(ID:=[ID]);
```

- **[ID] = ponteiro produzido na instrução de OPEN;**

Fecha o arquivo indicado pelo parâmetro [ID].

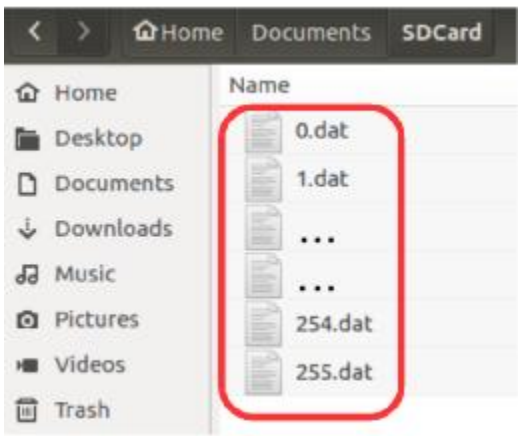
```
FILE_DELETE(NAME:=[NAME]);
```

- **[NAME] = *USINT* com valor como nome do arquivo;**

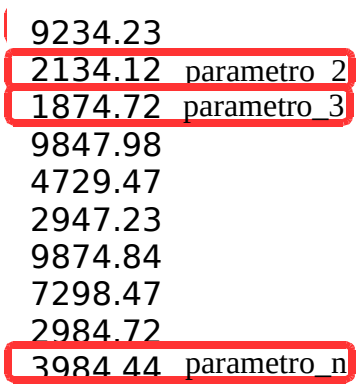
A instrução DELETE exclui o arquivo indicado pelo parâmetro [NAME].

Exemplo de código para registro histórico de 4 leituras, onde cada registro ocupa uma linha e um novo registro é lançado a cada minuto:

```
TimerCard();
IF TimerCard.CV>=T#1m THEN
        FILE_OPEN(NAME:=1502,ID=>ponteiro,ERROR=>ErroCartao);
        FILE_WRITE(ID:=ponteiro,DATA:=GP3220_RHOR);
        FILE_WRITE(ID:=ponteiro,DATA:=GP3220_RMIN);
        FILE_WRITE(ID:=ponteiro,DATA:=GP3220_SOMAn);
        FILE_WRITE(ID:=ponteiro,DATA:=GP3220_CPICOf);
        FILE_WRITE(ID:=ponteiro, NEW_LINE);
        FILE_CLOSE(ID:=ponteiro);
        TimerCard(^R:=1);
END_IF;
```



Nomenclatura dos arquivos no cartão



Exemplo de formação de um arquivo ".dat" com N parâmetros

4.4.5. Função PID

Para utilizar os algoritmos de PID disponíveis no Solver é necessário primeiramente realizar a instanciação do bloco de função de tipo PID_A. Para implementar isso dentro de um POU, usa-se uma variável com nome escolhido pelo usuário e declarando o tipo adequado:

```
VAR
  Mpid: PID_A;
END_VAR
```

Depois de realizar a declaração da instância do PID, a função será chamada uma ou mais vezes para inicialização, passagem de parâmetros e aquisição de resultados. O exemplo a seguir implementa todos os recursos em uma única chamada:

```
Mpid(EN:=TRUE, RST:=PIDresetB, KP:=KP01, KI:=KI01, KD:=KD01, ^T:=T#1s,
INV:=TRUE, LMAX:=1000.0, LMIN:=0.0, PV:=MD01, SP:=SP01, CO=>PIDsaida,
PRESET_VALUE:=PIDpreset);
```

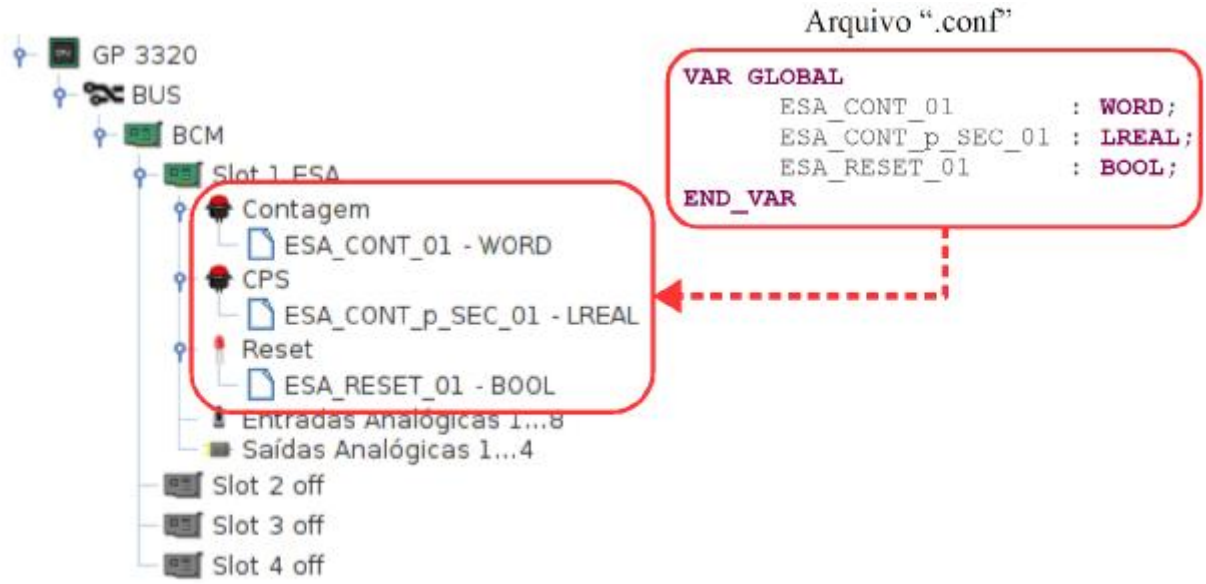
Significado dos parâmetros:

- *EN*: (BOOL) Habilita a operação da função PID;
- *RST*: (BOOL) Reinicia os registros internos para colocar a saída no valor definido pelo parâmetro *PRESET_VALUE;*
- *KP*: Ganho proporcional da malha PID;
- *KI*: Ganho integral da malha PID;
- *KD*: Ganho derivativo da malha PID;
- *^T*: Período de execução da função PID. Normalmente deve ser o mesmo valor do período da Task. Usado para compensação dos termos integral e derivativo;
- *INV*: (BOOL) Define o sinal do erro (direção da saída). Quando TRUE, a saída aumenta quando o setpoint (SP) for maior que o valor medido (PV). Quando FALSE, a saída atua na direção contrária;
- *LMAX*: Limite superior para a variável de saída;
- *LMIN*: Limite inferior para a variável de saída;
- *PV*: Valor medido da grandeza de controle (present value);
- *SP*: Setpoint da grandeza de controle;
- *CO*: Valor de saída da função PID;
- *PRESET_VALUE*: Valor forçado na variável de saída quando executado o reset (*RST*);

Parâmetros omitidos na chamada da função assumem o valor zero (ou False, se booleano).

4.4.6. Leitura de encoder

O GP3320 permite a leitura de sinais de encoder através do módulo GP3ESA. A leitura e processamento dos sinais do encoder são feitos por um hardware exclusivo no módulo - dessa forma, a velocidade de resposta não é influenciada pelo tempo de varredura do programa do usuário. Para utilizar a leitura de encoder é necessário primeiramente declarar as variáveis globais correspondentes. Na figura a seguir podemos observar a declaração do arquivo ".conf", destacando o mapeamento destas variáveis:



Mapeamento das variáveis de encoder na placa ESA

Com o mapeamento das variáveis realizado, pode-se agora utilizá-las no desenvolvimento do código de automação. No trecho de código seguinte é possível observar a forma como essas variáveis são empregadas, por exemplo, em linguagem ST:

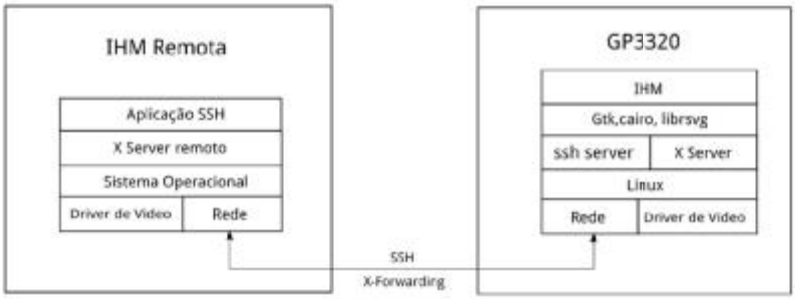
```
VAR_EXTERNAL
      // ESA1 - CTR1
      ESA_CONT_01       : WORD;  // Contador de 16 bits da CTR
      ESA_CONT_p_SEC_01 : LREAL; // Contagens por segundo
      ESA_RESET_01      : BOOL;  // Reset da contagem de Ct1
END_VAR
```

Utilizando as variáveis de leitura do encoder na linguagem ST

O valor de contagem é obtido a partir da diferença de fase do sinal SEN A em relação ao SEN B na entrada do módulo GP3ESA. Quando o sinal SEN A sobe antes do sinal SEN B, o valor da variável é incrementado (neste exemplo é *ESA_CONT_01*), e de maneira similar, quando o movimento inverso ocorre (SEN A subindo depois de SEN B) o valor é decrementado.

4.4.7. Servidor de telas remoto

O Servidor de telas remoto tem a função de disponibilizar a visualização de telas da IHM em quaisquer dispositivos que preencham os requisitos descritos abaixo:



Camadas de software para visualização de uma IHM remota

No GP3320 a IHM utiliza as bibliotecas Gtk3, Cairo e librsvg para desenhar a tela gráfica. Estas bibliotecas se comunicam com o XServer por meio de um protocolo específico que envia comandos de desenho. O XServer, por sua vez, recebe estes comandos e se comunica com o Kernel Linux para utilizar o driver gráfico.
Para a IHM remota a máquina terminal deve ter instalado uma aplicação SSH com suporte à X-Forwarding. Neste caso o terminal remoto irá se conectar com o GP3320 por meio do protocolo SSH, requisitando a execução de uma instância do programa de IHM. Para esta nova instância, o SSH server do GP3320 utilizará a rede para enviar comandos de desenho para o XServer presente na máquina remota.
O software de SSH utilizado para testes no Windows foi o MobaXTerm (https://mobaxterm.mobatek.net/), este software é gratuito e conta com SSH com X-Forwarding e XServer embutido, precisando apenas deste software para iniciar uma IHM remota. Seguem as instruções para executar a IHM remota no Windows:

1) **Baixar e instalar o software MobaXTerm do site (http://mobaxterm.mobatek.net/download.html).**

2) **Conectar o computador e o controlador na mesma rede IP;**

3) **No menu superior do MobaXTerm entrar em "Sessions" e clicar em "New Session";**

4) **Selecionar a opção SSH;**

5) **Ir na aba "Advanced SSH" e marcar a opção "X11-Forwarding" e "Compression";**

6) **Em "Execute command: " escrever "`ihm_gtk /bcm`";**

7) **Preencher a aba "remote host" com o IP do controlador;**

8) **Clicar no botão "ok";**

9) **Quando requisitado informar o usuário "`root`" e senha "`bcmbcm`";**

Para a visualização remota em uma máquina Linux, deve-se utilizar o cliente SSH nativo e o software de XServer remoto Xephyr.

4.4.8. Monitor de tempos de execução

O controlador programável GP3320 oferece recursos para medição dos tempos de execução efetivos das TASKS configuradas bem como também dos tempos dos ciclos de tratamento dos módulos de E/S locais e remotos. Estes recursos tem duas finalidades básicas:

- **Estimar o tempo de resposta do controlador a partir de um estímulo externo ou operação**
- **Monitorar e garantir que as periodicidades previstas para as Tasks estão sendo atendidas**

Todas as medidas de tempo tem resolução de 100us. Para ter acesso às medidas, deve ser incluir no programa do usuário as funções específicas que carregam estas medidas nas variáveis escolhidas pelo programador. Uma vez no programa do usuário, estas variáveis podem ser usadas para apresentação no display, envio a outros equipamentos, inclusão em lógicas de alarme, etc...

Função (FUNCTION) *GETTASKID*

Descrição: Converte o nome da Task em um identificador para uso subsequente no bloco de função TASK_INFO.

Parâmetros:

NAME
Função: Nome da task;
Tipo: Texto;
Direção: Entrada
Categoria: Obrigatório;

ID
Função: Identificador da Task;
Tipo: UINT
Direção: Saída;
Categoria: Obrigatório:

ERROR
Função: Indicador de erro da operação. Por exemplo, nome da Task inválido;
Tipo: BOOL;
Direção: Saída;
Categoria: Opcional;

Bloco de função (FUNCTION_BLOCK) *TASK_INFO*

Descrição:
Coleta os dados referentes à estatística de uso de uma Task;

Parâmetros:

Identificador
Função: Identificador da Task;
Tipo: UINT;
Direção: Entrada;

Estado
Função: Informa o estado da Task (0:TASK_WAITING, 1:TASK_RUNNING, 2:TASK_STOPPED, 3:TASK_FREE_RUNNING, 255:TASK_ERROR);
Tipo: USINT;
Direção: Saída;

TaxaUso

Função: Informa o processamento ocupado pela Task como um percentual do processamento total ocupado pelas lógicas de automatismo;

Tipo: REAL;

Direção: Saída;

Periodo

Função: Informa o período instantâneo de execução da Task;

Tipo: TIME;

Direção: Saída;

Observação: O tempo é representado em nanosegundos;

PeriodoMinimo

Função: Informa o período mínimo de execução da Task;

Tipo: TIME;

Direção: Saída;

Observação: O tempo é representado em nanosegundos;

PeriodoMaximo

Função: Informa o período máximo de execução da Task;

Tipo: TIME;

Direção: Saída;

Observação: O tempo é representado em nanosegundos;

PeriodoMedio

Função: Informa o período médio de execução da Task;

Tipo: TIME;

Direção: Saída;

Observação: O tempo é representado em nanosegundos. O valor médio é calculado por um filtro digital de primeira ordem;

Tempo

Função: Informa o tempo instantâneo (duração) da Task;

Tipo: TIME;

Direção: Saída;

Obervação: O tempo é representado em nanosegundos;

TempoMinimo

Função: Informa o tempo mínimo instantâneo (duração) da Task;

Tipo: TIME;

Direção: Saída;

Obervação: O tempo é representado em nanosegundos;

TempoMaximo

Função: Informa o tempo máximo instantâneo (duração) da Task;

Tipo: TIME;

Direção: Saída;

Obervação: O tempo é representado em nanosegundos;

TempoMedio

Função: Informa o tempo médio (duração) da Task;

Tipo: TIME;

Direção: Saída;

Obervação: O tempo é representado em nanosegundos. O valor médio é calculado por um filtro digital de primeira ordem;

Atividade

Função: Contador do número de execuções da Task;
Tipo: ULINT;
Direção: Saída;

SobreCarga

Função: Contador do número condições de sobrecarga da Task;
Tipo: ULINT;
Direção: Saída;
Observação: Um evento de sobrecarga é considerado quando a condição de execução da tasks é habilitada (período ou evento) sem que o ciclo anterior da task tenha sido completado.

Disparos

Função: Contador do número de disparos (trigger) da task por eventos;
Tipo: ULINT;
Direção: Saída;
Observação: A condição de disparo ocorre em uma transição de FALSE para TRUE da variável SINGLE da Task;

UsoPilha

Função: Registra o número máximo de bytes utlizados na pilha interna da Task;
TIPO: ULINT;
Direção: Saída;
Observação: A pilha interna de cada Task possui 32Kbytes e é utilizada para variáveis internas de funções e armazenamento de endereços de retorno na chamada de FUNCTIONS e FUNCTIONS_BLOCKS;

Single

Função: Mostra o valor instantâneo do parametro SINGLE da Task;
Tipo: BOOL;
Direção: Saída;

Interval

Função: Mostra o valor instantâneo do parametro INTERVAL da Task;
Tipo: TIME;
Direção: Saída;

Reset

Função: Reset dos contadores e registro de valor máximo da Task;
Tipo: BOOL;
Direção: Entrada;

Bloco de função (FUNCTION_BLOCK) RESOURCE_INFO

Descrição:

Coleta os dados referentes a estatistica de uso de um Resource;

Parametros:

UsoCpu

Função: Informa o processamento ocupado pela automatismo como um percentual do processamento total disponível do CLP;
Tipo: REAL;
Direção: Saída;

Bloco de função (FUNCTION_BLOCK) BUS_INFO

Descrição:

Coleta os dados referentes a estatistica de uso de um barramento de campo;

Parametros:

Identificador

Função: Identificador do barramento (0:BCM_BUS, 1:ETHERCAT);
Tipo: USINT;
Direção: Entrada;

Periodo

Função: Informa o periodo instantaneo de SCAN do barramento;
Tipo: TIME;
Direção: Saída;
Observação: O tempo é representado em nanosegundos;

PeriodoMinimo

Função: Informa o periodo mínimo registrado de SCAN do barramento;
Tipo: TIME;
Direção: Saída;
Observação: O tempo é representado em nanosegundos;

PeriodoMaximo

Função: Informa o periodo máximo registrado de SCAN do barramento;
Tipo: TIME;
Direção: Saída;
Observação: O tempo é representado em nanosegundos;

PeriodoMedio

Função: Informa o periodo médio de SCAN do barramento;
Tipo: TIME;
Direção: Saída;
Observação: O tempo é representado em nanosegundos. O valor médio é calcula por um filtro digital de primeira ordem;

Duracao

Função: Informa o tempo instantaneo (duração) do SCAN do barramento;
Tipo: TIME;
Direção: Saída;
Observação: O tempo é representado em nanosegundos;

DuracaoMinimo

Função: Informa o tempo mínimo registrado (duração) de SCAN do barramento
Tipo: TIME;
Direção: Saída;
Obervação: O tempo é representado em nanosegundos;

DuracaoMaximo

Função: Informa o tempo máximo registrado (duração) de SCAN do barramento;
Tipo: TIME;
Direção: Saída;
Observação: O tempo é representado em nanosegundos;

DuracaoMedio

Função: Informa o tempo médio de SCAN do barramento;

Tipo: TIME;

Direção: Saída;

Obervação: O tempo é representado em nanosegundos. O valor médio é calcula por um filtro digital de primeira ordem;

Atividade

Função: Contador do número de SCAN do barramento;

Tipo: ULINT;

Direção: Saída;

Reset

Função: Reset dos contadores e registro de valor máximo do tempo de varredura do barramento;

Tipo: BOOL;

Direção: Entrada;

4.4.8.1. Passo a passo para implementar os recursos de monitoração:

a) Dentro do POU escolhido para acolher as funções de monitoração, declarar as variáveis que irão receber as medidas desejadas, com nomes livres e tipos compatíveis com os valores que irão assumir. Normalmente são declarados na área VAR_EXTERNAL. Lembrar que as mesmas variáveis devem ser declaradas também no arquivo de configuração (.CONF) como variáveis globais.

b) Na área VAR, criar uma ou mais instâncias do bloco de função TASK_INFO, responsável por adquirir as medidas. Exemplo:

statusTask : TASK_INFO; //statusTask é um nome criado pelo usuário.

c) Obter o identificador da primeira Task desejada usando o bloco de função GETTASKID. Exemplo para a Task denominada T2:

GETTASKID(NAME:=T2, ID=>id); //id é uma variável criada pelo usuário, deve ser INT.

d) Chamar a função que irá buscar as medidas, passando o identificador da task desejada:

statusTask(Identificador:=id);

e) A seguir, cada medida pode ser passada da função para a variável que o usuário deseja. Exemplo:

Estado_0 := statusTask.Estado;
periodoMaximo_0 := statusTask.PeriodoMaximo/ULINT#1000000.0; //período sai em ms!

Uma alternativa é passar a medida na mesma linha em que chama a função, na forma:

statusTask(Identificador:=id, Estado=>Estado_0);

f) Repetir os itens 'c', 'd', 'e' para todas as tasks desejadas.

Para incluir monitoração do ciclo de atendimento de E/S local:

g) Na área VAR, criar uma instância do bloco de função BUS_INFO, responsável por adquirir as medidas. Exemplo:

statusES : BUS_INFO; //statusES é um nome criado pelo usuário.

h) Chamar a função que irá buscar as medidas, passando o identificador que especifica que a medição será relacionada ao barramento de E/S local:

statusES(Identificador:=0);

i) Chamar a função que irá buscar as medidas, passando o identificador que especifica que a medição será relacionada ao barramento de E/S local:

statusES(Identificador:=0);

j) A seguir, cada medida pode ser passada da função para a variável que o usuário deseja. Exemplo:

periodoMaximoES := statusES.PeriodoMaximo/1000000.0;

Uma alternativa é passar a medida na mesma linha em que chama a função, na forma:

statusES(Identificador:=0, PeriodoMaximo=>periodoMaximoES);

4.5. A Linguagem Descritiva para o GP3320

A programação em Linguagem Descritiva para o controlador GP3320 segue a estrutura e diretrizes descritas no manual: *Linguagem Descritiva BCM* (31940092-8). No entanto, algumas diferenças importantes devem ser consideradas:

a) A programação do display é feita por um módulo próprio do Solver - além disso, no GP3320 não há distinção entre modo gráfico e modo alfanumérico no display. Em função disso, a instrução *MOSTRA* não é mais aplicável, e as malhas usadas para gerenciamento da navegação do display devem ser reestruturadas numa aplicação para o GP3320.

b) A comunicação via canais seriais e Ethernet é configurada numa área especifica do Solver, onde são descritas as tarefas, mapeamento de variáveis, endereços e demais parâmetros. Em função disso, as instruções *LE* e *ESCREVE* não são mais aplicáveis num projeto para o GP3320.

4.5.1) Instruções suportadas pelo GP3320 para a Linguagem Descritiva

MALHA	***FACA***	***SET***
ESTADO	***SE***	***RESET***
LIGA	***VA PARA***	***INCREMENTA***
DESLIGA	***COPIA***	***DECREMENTA***

4.5.2) Diferenças de sintaxe na programação GP3320 com relação aos demais modelos

a) O referenciamento das malhas no GP3320 deve ser feito por um nome (alfanumérico), ao invés de um número. Exemplo: *MALHA Controle_Caldeira:*

b) Exceto os comandos *MALHA* e *ESTADO*, que terminam com ":", todas as demais linhas devem ser terminadas com ";"

c) No Solver, os comentários devem ser precedidos pelo sinal "//", ao invés do ";" usado nos demais modelos.

d) Nas instruções *FACA*, as atribuições devem ser sinalizadas com ":=", ao invés do "=" usado nos demais modelos.

e) A precedência das operações matemáticas no GP3320 segue a ordem padrão da maioria das linguagens (multiplicação/divisão primeiro, soma/subtração depois) e não mais a ordem em que as operações aparecem na linha.

f) O valor a ser lançado numa linha *SE ATRASO=...* deve ser uma variável do tipo *TIME* ou uma constante caracterizada como tipo *TIME* (por exemplo: *T#30s*), ao invés do valor numérico simples (em 1/10s) usado nos demais modelos.

g) Na instrução *FACA*, além das operações básicas disponíveis para o GP3220, podem ser usadas todas as demais funções e operadores descritos no item 5.2 deste manual, além de funções eventualmente desenvolvidas pelo usuário.

h) Todas as palavras chave devem ser escritas exclusivamente em letra maiúscula.