

AGT DISPLAYS

Modbus no Módulo Serial

O display falando direto com PLC e instrumento, como mestre ou como escravo, sem MCU no meio

PARA QUEM

Quem precisa ler um PLC ou instrumento sem escrever firmware

COMO SE PROGRAMA

Por tabela no arquivo 22, não por código

PRÉ-CONDIÇÃO

Módulo com UART4 exposta — não é universal

ATENÇÃO

Existem duas implementações distintas — ver §3

1. O QUE ISSO RESOLVE

O arranjo comum de uma aplicação com display serial tem três peças: o display, um MCU e o equipamento. O MCU existe para traduzir — ele fala o protocolo do display de um lado e o protocolo do equipamento do outro.

Quando o equipamento é um PLC, um inversor, um medidor ou um controlador de temperatura, esse protocolo quase sempre é Modbus. E aí o MCU intermediário passa a ser um tradutor entre dois protocolos que o próprio display sabe falar — trabalho que existe só porque ninguém ligou os dois direto.

Com o suporte a Modbus habilitado, o display conversa com o equipamento diretamente. Um valor lido do PLC aparece num endereço de variável do projeto e é exibido por um controle comum, como qualquer outro dado. Um valor digitado na tela é escrito no registrador do PLC. Não há código de aplicação envolvido: a comunicação é descrita numa **tabela de instruções**, e o módulo a executa em laço.

É configuração, não programação. Toda a comunicação se descreve preenchendo linhas de uma tabela: com quem falar, qual função Modbus usar, quantos dados, de onde para onde, e quando disparar. Quem já montou tabela de pontos em software de supervisão vai reconhecer o modelo imediatamente. Quem esperava escrever código vai achar rígido — e é: o que a tabela não expressa, não acontece.

2. A PRÉ-CONDIÇÃO DE HARDWARE

O Modbus roda na **UART4**. Isso não é configurável nas implementações documentadas – a porta de saída é fixa.

E aqui está a primeira coisa a verificar, antes de qualquer decisão de projeto: **nem todo módulo expõe a UART4**. Muitos modelos trazem apenas a UART2, que é a porta de comunicação padrão com o MCU. Em outros, a UART4 existe no ASIC mas não chega a conector – fica em pads de solda, ou é reservada para outra função na placa.

Confirme três coisas com o suporte da AGT Displays, com o modelo em mão, antes de especificar Modbus no projeto:

- **A UART4 está acessível** nesse modelo, e em conector ou em pad de solda.
- **Em que nível elétrico**. A UART4 pode sair em RS485, RS232 ou TTL, conforme o modelo. Qualquer um dos três serve ao Modbus – veja abaixo o que cada um permite.
- **Se a UART2 continua livre**. Se o projeto ainda precisa de um MCU para outra coisa, as duas portas convivem; se a UART2 for a única porta do modelo, Modbus e MCU disputam a mesma linha.

2.1. RS485 NÃO É EXIGÊNCIA

O Modbus RTU é um protocolo de quadro: define bytes, endereços e CRC, não a camada elétrica. O módulo monta e interpreta os mesmos quadros qualquer que seja o nível da porta. RS485 é o mais comum em instalação industrial porque é o único dos três que atende a barramento longo com vários equipamentos – mas RS232 e TTL funcionam, dentro dos limites de cada um:

NÍVEL	TOPOLOGIA	DISTÂNCIA	QUANDO FAZ SENTIDO
RS485	Barramento: um mestre e vários escravos na mesma linha de dois fios	Longa – centenas de metros com cabo par trançado	Painel industrial, vários equipamentos, cabo que atravessa a planta
RS232	Ponto a ponto: um mestre e um escravo	Curta – poucos metros	Um único equipamento próximo com porta RS232 (PLC compacto, inversor, instrumento de bancada)
TTL	Ponto a ponto, sem isolamento nem imunidade a ruído	Muito curta – mesma placa ou cabo de poucos centímetros	Outro circuito dentro do mesmo equipamento: um MCU, um módulo de medição, uma placa de expansão

Duas consequências práticas. Com RS232 ou TTL o módulo conversa com **um equipamento só**: a tabela de mestre pode ter muitas linhas, mas todas para o mesmo endereço de escravo. E precisar de alcance ou de mais de um escravo numa porta que sai em RS232 ou TTL não obriga a trocar de módulo – resolve-se com um conversor para RS485 externo, sem mudar nada na configuração.

Se a porta for RS485: meio-duplex. Numa linha de dois fios, ninguém transmite enquanto recebe. Isso já está tratado dentro do módulo, mas importa na instalação: terminação nas duas extremidades do barramento, malha aterrada em um ponto só, e derivação curta. Barramento Modbus que funciona na bancada e falha no painel é, quase sempre, instalação e não configuração.

3. EXISTEM DUAS IMPLEMENTAÇÕES – DESCUBRA QUAL É A SUA

Esta seção vem antes de qualquer parâmetro por um motivo prático: a documentação do fabricante descreve **duas implementações de Modbus diferentes**, com blocos de configuração em posições diferentes, formatos de instrução de tamanhos diferentes e limites diferentes. Preencher a tabela de uma delas num módulo que roda a outra não funciona e não dá mensagem de erro.

	VARIANTE A – NÚCLEO MODBUS DEDICADO	VARIANTE B – PLATAFORMA DE DESENVOLVIMENTO
Como chega	Arquivo de núcleo pronto, fornecido pelo fabricante (mestre ou escravo, conforme o papel)	Recurso da plataforma de desenvolvimento secundário; exige núcleo de OS V21 ou superior
Bloco de parâmetros	Interface de variáveis de sistema, 16 bytes a partir de 0x0088	Espaço de variáveis, a partir de 0xE000
Tabela de instruções	No arquivo 22, a partir do endereço de word configurado (padrão 0xE000)	No arquivo 22, de 0x1C000 a 0x1FFFF (16 KB); instruções do usuário a partir de 0x1C010
Tamanho da linha	8 words	16 bytes
Máximo de linhas	255	1023
Máximo por instrução	64 words (128 bytes)	180 bytes
Modos de disparo	Cinco (0 a 4)	Três (0 a 2)
ID de escravo padrão	0x01	0x005A
Baud	Divisor: 25804800 / baud	Valor direto em Kbps com uma decimal

Antes de preencher uma linha de tabela, confirme a variante. As duas são legítimas e ambas estão documentadas pelo fabricante. O que determina qual vale é o arquivo de núcleo que está gravado no seu módulo. Peça ao suporte da AGT Displays o arquivo de núcleo para o seu modelo e o papel desejado, e trabalhe com a seção correspondente deste documento – §5 para a variante A, §6 para a variante B.

O que é **comum às duas** e vale sempre: a porta é a UART4; o formato é RTU; o papel de mestre ou escravo é escolhido na configuração; a tabela vive no arquivo 22; o dado trafega pelo espaço de variáveis do projeto e é exibido por controles comuns.

4. MESTRE OU ESCRAVO

A escolha não é de gosto, é de topologia: define quem pergunta.

4.1 Display como mestre

O display é quem interroga. Ele percorre a tabela de instruções, envia cada pergunta ao equipamento, recebe a resposta e deposita o valor no endereço de variável indicado. É o arranjo para **display supervisionando equipamento**: ler um PLC, escrever setpoint num inversor, acompanhar um medidor.

É o caso mais comum e o que exige mais configuração, porque cada dado que você quer ver na tela é uma linha de tabela.

4.2 Display como escravo

O display atende. Um mestre externo – PLC, supervisório, PC – escreve nos endereços de variável do projeto e lê de volta o que o operador digitou. O display não toma iniciativa nenhuma.

É o arranjo para **display como painel de um sistema maior**, onde o PLC já é o mestre da linha e o display é mais um nó. A configuração é muito mais simples: não há tabela de instruções a preencher, porque não há perguntas a fazer – basta declarar o papel, o ID e os parâmetros de linha.

Um papel por vez. O módulo é mestre ou escravo, não os dois. E, na variante A, o papel vem decidido no próprio arquivo de núcleo: trocar de papel significa gravar o outro núcleo, removendo antes o anterior. Decida a topologia no início do projeto.

5. VARIANTE A – NÚCLEO MODBUS DEDICADO

A configuração ocupa 16 bytes da interface de variáveis de sistema, a partir de **0x0088**.

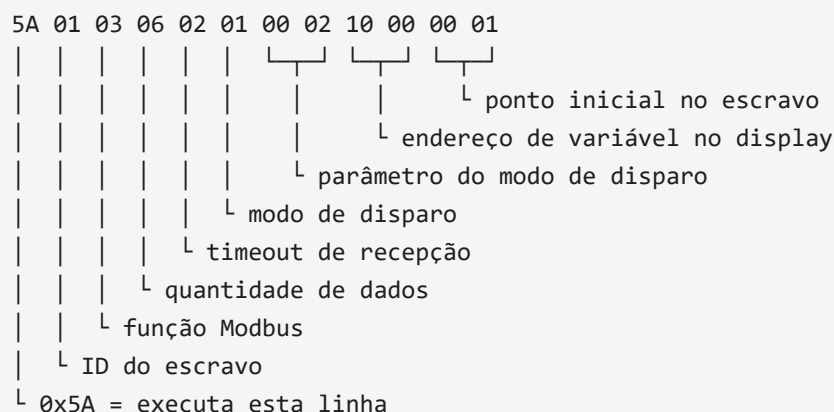
POSIÇÃO	O QUE É	VALORES
0x0088:H	Dispara a aplicação da configuração	0x5A; zera depois de aplicada
0x0088:L	Habilita o Modbus	0xA5 habilita (padrão); outro valor desabilita
0x0089	Endereço de word da primeira instrução no arquivo 22	Padrão 0xE000
0x008A:H	Quantidade de instruções na tabela	1 a 255; padrão 0xFF
0x008A:L	Papel e formato	0x00 mestre RTU (padrão); 0x02 mestre RTU com relógio na resposta; 0x04 escravo RTU
0x008B:H	Retransmissões quando o escravo não responde ou responde errado	0 a 255; padrão 5
0x008B:L	Intervalo entre instruções, em ms	0 a 255; padrão 0x64 = 100 ms
0x008C:H	ID próprio, quando o display é escravo	0x01 a 0x7F; padrão 0x01
0x008C:L	Paridade e porta	Padrão 0x40 – nibble alto 4 = UART4, nibble baixo 0 = 8N1 (1 = 8E1, 2 = 8O1, 3 = 8N2)
0x008D	Divisor de baud	25804800 / baud. Para 9600: 2688 = 0x0A80
0x008E:H	Código de falha da comunicação	Somente leitura
0x008E:L	Número da instrução em execução	Somente leitura; avança quando o escravo responde certo

Exemplo de bloco completo, para mestre RTU em 9600 8N1, tabela a partir de 0xE000 com 255 linhas:

```
5A A5 E000 FF 00 05 64 01 40 0A80 0000 0000
```

5.1 A linha de instrução

Cada instrução ocupa 8 words. Exemplo: ler 6 words consecutivas do escravo 0x01, começando no ponto 0x0001, para os endereços 0x1000 a 0x1005 do display:



CAMPO	SIGNIFICADO
D0	0x5A executa esta linha; qualquer outro valor a ignora. É o jeito de desabilitar uma linha sem apagá-la.
D1	ID do escravo a ser operado.
D2	Função Modbus: 0x01, 0x02, 0x03, 0x04, 0x05, 0x06 ou 0x10.
D3	Quantidade de dados. Nas funções 0x03, 0x04, 0x06 e 0x10, contada em words.
D4	Timeout de recepção, em unidades de 2 ms. Faixa 2 a 255.
D5	Modo de disparo — ver §5.2.
D6.D7	Parâmetro do modo: número de página no modo 1, endereço da tecla no modo 2, não usado nos demais.
D8.D9	Endereço de variável do display: destino numa leitura, origem numa escrita.
D10.D11	Endereço inicial na tabela de pontos do escravo.
D12-D15	Não definidos.

5.2 Os cinco modos de disparo

Este campo é o que separa uma configuração que funciona de uma que sobrecarrega o barramento. Ele decide *quando* a linha é executada.

MODO	QUANDO EXECUTA	PARA QUE SERVE
0x00	Sempre, incondicionalmente	Dado que precisa estar atualizado independente da tela mostrada – alarme, estado de máquina
0x01	Só quando a página indicada está na tela	O modo mais útil. Não gasta barramento lendo o que ninguém está vendo
0x02	Quando a tecla indicada tem valor diferente de zero; a tecla é zerada depois	Ação sob comando – botão "enviar", "confirmar", "acionar"
0x03	Na função 0x06, quando a variável de origem é diferente de zero; zerada após a comunicação	Escrita pontual disparada por valor
0x04	Nas funções 0x05, 0x06 e 0x10, quando a variável de origem muda de valor	Escrita automática: o operador digita, o valor vai. Tem retransmissão e não exige confirmação de leitura

Use o modo 1 por padrão nas leituras. A tentação é deixar tudo em modo 0, porque sempre funciona. O resultado é um barramento ocupado relendo dezenas de pontos que estão em telas fechadas, com cada leitura somando o intervalo entre instruções – e uma tela que atualiza devagar justamente porque tudo está sendo lido. Amarre cada leitura à página onde ela aparece.

6. VARIANTE B – PLATAFORMA DE DESENVOLVIMENTO

Nesta variante os parâmetros ficam no espaço de variáveis a partir de **0xE000**, com a inicialização no arquivo 22 entre **0x1C000** e **0x1FFFF**. Exige núcleo de OS **V21 ou superior**.

POSIÇÃO	O QUE É	VALORES
0xE000	Marca de execução	0x5AA5 para rodar o Modbus
0xE001	Papel	0x0000 mestre; qualquer outro valor, escravo
0xE002	Baud	Em Kbps com uma decimal. 115200 bps = 0x480. Se 0x0000, volta ao padrão de 115200
0xE003	Formato serial	0x0000 8N1; 0x0001 8E1; 0x0002 8O1; 0x0003 8N2; outro = 8N1
0xE004	ID próprio no modo escravo	Padrão 0x005A
0xE005:H	Porta de saída	Somente leitura. Padrão 0x04 = UART4
0xE005:L	Versão do OS	Somente leitura
0xE008+	Primeira linha de instrução	16 bytes por linha – ver abaixo

Precondição: a inicialização pelo arquivo 22 tem de estar habilitada. O byte de configuração de sistema na posição **0x08** do arquivo de configuração de hardware carrega no **bit .5** o habilitador da inicialização pelo arquivo 22 – 1 liga, 0 desliga. O valor de fábrica desse byte é **0x38**, que já tem o bit .5 em 1. Se o módulo parecer ignorar tudo o que você escreveu no arquivo 22, confira esse bit antes de qualquer outra coisa. A documentação da variante B declara essa precondição explicitamente; a da variante A não a menciona, embora também guarde a tabela no arquivo 22.

6.1 A linha de instrução

OFFSET	SIGNIFICADO
0x00	0x5A = linha válida; outro valor = inválida
0x01	ID do dispositivo Modbus a ler ou escrever
0x02	Função Modbus
0x03	Comprimento dos dados. 0x00 invalida a linha
0x04	Tempo da instrução em ms, até 9999. Numa leitura, é o tempo de resposta do escravo. Padrão 60 ms; 0x0000 usa o padrão
0x06	Modo de execução: 0x0000 em todas as páginas; 0x0001 só na página indicada; 0x0002 quando o byte baixo da variável indicada é 0x5A, limpando-a depois
0x08	Número da página no modo 1; endereço de variável no modo 2
0x0A	Endereço inicial no espaço de variáveis, de 0x0000 a 0xDFFF
0x0C	Endereço inicial no dispositivo Modbus
0x0E	Estado da comunicação: 0x0000 falhou, 0x00FF sucesso. Deve ser limpo após a leitura

Regras próprias desta variante: a faixa de variáveis acessível é 0x0000 a 0xDFFF, e a região abaixo de 0x1000 é a interface de variáveis de sistema; uma linha não pode transferir mais de 180 bytes, nem terminar acima de 0xDFFF — nos dois casos a linha é ignorada e a execução segue para a próxima. Com o baud no padrão, o fabricante sugere 20 ms de tempo de instrução.

6.2 Como isso fica no arquivo

Configuração de **mestre**, como aparece no arquivo 22 aberto num editor hexadecimal. A primeira linha é o bloco de parâmetros; da segunda em diante, uma instrução por linha de 16 bytes:

```
0001C000: 5A A5 00 00 04 80 00 00 00 5A 00 00 00 00 00 00
0001C010: 5A 01 03 14 01 F4 00 00 00 00 14 00 00 00 00 00
0001C020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001C030: 5A 01 10 14 01 F4 00 00 00 00 15 00 00 0A 00 00
0001C040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001C050: 5A 01 03 0A 01 F4 00 02 16 00 17 00 00 14 00 00
0001C060: 5A 01 10 0A 01 F4 00 02 16 01 17 10 00 20 00 00
0001C070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001C080: 5A 01 03 02 01 F4 00 01 00 02 17 20 00 25 00 00
```

Linha por linha:

ENDEREÇO	O QUE ESSA LINHA FAZ
1C000	Parâmetros: 5A A5 executa, papel 0x0000 = mestre, baud 0x0480 = 115200, formato 8N1, ID próprio 0x005A – sem efeito no papel de mestre
1C010	Lê 20 words do escravo 0x01 (função 0x03), 500 ms, em todas as páginas, do ponto 0x0000 para o endereço 0x1400
1C030	Escreve 20 words (função 0x10) de 0x1500 para o ponto 0x000A, em todas as páginas
1C050	Lê 10 words para 0x1700, modo 2 : só executa quando o byte baixo de 0x1600 for 0x5A, limpando-o depois
1C060	Escreve 10 words de 0x1710 para o ponto 0x0020, disparada por 0x1601
1C080	Lê 2 words para 0x1720, modo 1 : só na página 0x0002

Note as linhas zeradas em 1C020, 1C040 e 1C070. Elas não são espaçamento: são **linhas inválidas**, porque o primeiro byte não é 0x5A e o comprimento é zero. O módulo as ignora e segue para a próxima. Deixar espaço entre grupos de instruções é legítimo e não custa nada.

A mesma região configurada como **escravo** é bem mais curta – só o bloco de parâmetros, com o papel diferente de zero e nenhuma instrução:

```
0001C000: 5A A5 00 01 04 80 00 00 00 5A 00 00 00 00 00 00
0001C010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001C020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

O 0x0001 no papel põe o módulo como escravo, e o 0x005A passa a valer como ID próprio. Daí em diante é tudo zero: quem pergunta é o mestre externo.

7. AS FUNÇÕES MODBUS SUPORTADAS

O conjunto é o clássico, e cobre praticamente todo equipamento industrial. Aqui há mais uma divergência entre as variantes: a variante A documenta sete funções, e a variante B documenta nove — as duas a mais são 0x07 e 0x0F. A coluna da direita diz qual variante documenta cada uma.

CÓDIGO	FUNÇÃO	COMPRIMENTO	VARIANTE
0x01	Lê estado de bobinas	Fixo em 16 bits na variante A; na B, número de bobinas dividido por 8	A e B
0x02	Lê estado de entradas discretas	Fixo em 16 bits na variante A; na B, número de bits dividido por 8	A e B
0x03	Lê registradores de retenção	Em words	A e B
0x04	Lê registradores de entrada	Em words	A e B
0x05	Escreve uma bobina	Um bit	A e B
0x06	Escreve um registrador	Uma word	A e B
0x07	Lê estado de exceção	Um byte; endereço inicial indiferente	Só B
0x0F	Escreve várias bobinas	Número de bobinas	Só B
0x10	Escreve vários registradores	Em words	A e B

A ausência de 0x0F na variante A tem consequência de projeto: sem ela, escrever um bloco de bobinas exige uma linha de tabela por bobina, com a função 0x05. Se a aplicação aciona muitos bits de uma vez, isso pesa na conta de linhas e no tempo de varredura — e é um argumento concreto na escolha da variante.

Duas notas de comportamento na variante A. As funções 0x06 e 0x10 seguem o formato padrão. As funções 0x03 e 0x04 existem em duas formas: a padrão e uma com relógio, em que a resposta traz **6 bytes de data e hora** — ano, mês, dia, hora, minuto, segundo, em hexadecimal — inseridos antes do código de verificação. Só use a forma com relógio se o equipamento do outro lado a implementar.

Quando o mestre externo pede um endereço fora da faixa acessível, a resposta volta com o **bit mais alto do código de função em 1** — é a sinalização padrão de erro do Modbus, e é o que se observa no analisador quando o endereço pedido não existe do lado do display.

8. ENDEREÇAMENTO DE BIT

Aqui está a parte que mais gera confusão, porque envolve uma limitação de arquitetura: **o display não tem variáveis de bit**. Toda a memória é organizada em words.

Para que as funções de bobina funcionem, a variante A define duas áreas de 256 bits no espaço de variáveis, cada uma ocupando 16 words:

ÁREA	FAIXA	USO
Área de exibição	0x0100 a 0x010F	Bits lidos do equipamento, para mostrar na tela
Área de teclado	0x0110 a 0x011F	Bits acionados na tela, para escrever no equipamento

Cada word cobre 16 bits, numerados do bit 0 ao 15. O bit 0 da área de exibição está em 0x0100, o bit 32 em 0x0102, e assim por diante. Na tela, esses bits são exibidos por controles de ícone por bit, e acionados por controles de toque comuns.

As funções de bit leem sempre 16 bits – arredonde os dois endereços para baixo, em múltiplos de 16. Como não há variável de bit de verdade, as funções 0x01 e 0x02 transferem uma word inteira. Se o ponto que interessa no equipamento é o 301, o endereço a declarar é 300, e você recebe de 300 a 315 de uma vez. O endereço na área de exibição segue a mesma regra: declare 0, não 1. O bit que você queria estará na posição correspondente dentro da word.

8.1 Acionar uma bobina pela tela

Do lado da tela, um bit da área de teclado é acionado por um controle de toque comum. O arranjo documentado usa o **Incremental Adjustment** configurado em modo de bit – o mesmo controle que normalmente incrementa um número, restrito a dois valores:

PARÂMETRO	VALOR	POR QUÊ
VP	0x0113	A word da área de teclado que contém o bit desejado
Modo	Bit, <i>Bit num</i> = 3	Opera um bit só, não a word inteira
Adjust method	++	Cada toque incrementa
Over limit Operation	Cycle	Ao passar do limite, volta ao início – é o que faz alternar
Adjusting step length	1	Passo unitário
Lower / Upper limit	0 e 1	Restringe a faixa aos dois estados do bit
Touch effect	Constant Setting	Realimentação visual do toque

Com limites 0 e 1, passo 1 e retorno cíclico, cada toque inverte o bit. No exemplo, o VP **0x0113** com bit 3 é o **bit 51 da área de teclado** — a quarta word da área, quarto bit. A linha de tabela que leva isso ao equipamento usa a função **0x05** com disparo por mudança de valor.

Há ainda uma diferença de convenção a considerar: os bits do Modbus são definidos a partir do bit menos significativo, e a organização de memória do módulo parte do mais significativo. Em leituras de bobina, confira a ordem na primeira montagem em vez de presumir — é mais rápido observar um bit conhecido do que deduzir o mapeamento.

9. O DISPLAY COMO ESCRAVO, NA PRÁTICA

No papel de escravo, não há tabela a preencher – o mestre externo endereça diretamente o espaço de variáveis do projeto. O que o display expõe é a própria memória de variáveis, e os controles da tela leem e escrevem nela como sempre.

Exemplos de quadros, na forma documentada para a variante B, com o ID do dispositivo em 0x5A:

Ler 2 words a partir de 0x1400

Envio: 5A 03 14 00 00 02 CC D0

Resposta: 5A 03 04 00 01 00 02 C0 F7

Escrever 0x0001 em 0x1400

Envio: 5A 06 14 00 00 01 40 D1

Resposta: 5A 06 14 00 00 01 40 D1

Escrever 2 words a partir de 0x1400

Envio: 5A 10 14 00 00 02 04 00 01 00 02 F8 8B

Resposta: 5A 10 14 00 00 02 49 13

A faixa endereçável é de 0x0000 a 0xDFFF e o limite por transação é de 180 bytes. Pedido fora da faixa ou acima do limite recebe resposta de erro.

O endereço Modbus é o endereço de variável do projeto. Isso simplifica e obriga a disciplina: o mapa de pontos que você entrega a quem integra o sistema é o mapa de variáveis do seu projeto DGUS. Mudar um endereço de variável depois de entregue quebra a integração do outro lado. Congele o mapa antes de publicar.

10. O QUE O MODBUS OCUPA

Habilitar Modbus não é gratuito: o recurso consome partes do módulo que passam a não estar disponíveis para outra coisa. Na variante A, o fabricante documenta quatro reservas.

- **70 words do espaço de variáveis**, imediatamente após a última instrução da tabela no arquivo 22.
Ao planejar o mapa de variáveis, deixe essa região livre.
- **As duas áreas de bit**, de 0x0100 a 0x011F.
- **Um temporizador do sistema e uma página de registradores** – relevante apenas se houver programa de aplicação próprio rodando no módulo, caso em que há disputa de recurso.
- **A UART4 inteira**. Ela deixa de estar disponível para qualquer outro uso.

Somando com a regra de separação de faixas do projeto, o planejamento de memória fica: variáveis do projeto na faixa usual, área de bit reservada, região de tabela e as 70 words seguintes intocadas, e espaço de descritores bem longe de tudo isso.

11. ROTEIRO DE PRIMEIRA SUBIDA

A ordem abaixo é feita para que cada passo falhe sozinho, em vez de você descobrir três erros somados no fim.

1. **Confirme a UART4** no modelo, com nível elétrico e conector – §2.
2. **Obtenha o núcleo correto** para o modelo e o papel, e confirme qual variante ele implementa – §3.
3. **Grave o núcleo** por cartão, com alimentação firme, e confirme que o módulo volta a ligar normalmente.
4. **Monte o projeto** com uma única página e um único valor na tela, ligado a um endereço de variável conhecido.
5. **Preencha uma linha só** na tabela: uma leitura, função 0x03, um word, do ponto mais simples do equipamento.
6. **Teste contra um simulador** no PC antes de ir ao equipamento real. Um mestre ou escravo Modbus de bancada elimina o PLC da equação.
7. **Observe a linha** com analisador ou com o software de teste. Ver o quadro saindo separa "o display não pergunta" de "o equipamento não responde".
8. **Só então amplie** a tabela, uma página de cada vez, amarrando cada leitura à sua página.

O lado do equipamento também se configura. Nenhum PLC fala Modbus por padrão. Três coisas precisam ser acertadas lá: o **formato de comunicação** (baud, paridade, bits de dados e de parada), o **número de estação** – o ID que o display vai endereçar – e o **timeout**. Em várias famílias de PLC isso não tem tela de configuração: faz-se dentro do próprio programa, movendo constantes para registradores especiais no primeiro ciclo de varredura. Display configurado corretamente contra PLC sem Modbus habilitado é a causa mais comum de barramento silencioso.

A tabela de pontos vem do manual do equipamento. O endereço que você escreve na linha de instrução não é o nome interno do elemento no PLC – é o endereço Modbus que o fabricante do equipamento atribuiu àquele elemento. O manual do PLC traz essa correspondência, tipicamente uma tabela por tipo de elemento: entradas, saídas, memórias internas, registradores de dados. Monte e confira essa tradução antes de preencher a tabela; endereço Modbus deduzido do nome do elemento erra quase sempre.

12. DIAGNÓSTICO

Sintomas desta área, na ordem em que vale testar.

SINTOMA	CAUSA PROVÁVEL	AÇÃO
Nenhum quadro sai da porta	Modbus não habilitado, ou tabela não aplicada	Confira a marca de habilitação e a de execução — §5 ou §6
Quadros saem, nada responde	Baud, paridade, A e B invertidos, ou ID errado	Confirme os quatro; comece invertendo A e B
Configuração parece ignorada	Tabela preenchida no formato da outra variante	§3 — confirme qual núcleo está gravado
Resposta com o bit alto da função em 1	Endereço pedido fora da faixa acessível	§7
Valor aparece, mas errado por um fator	Escala do equipamento, ou ordem de bytes	Confira a documentação do ponto no equipamento
Bit certo, posição errada	Endereço de bobina não arredondado para múltiplo de 16	§8
Tela atualiza devagar	Tudo em disparo incondicional, ou intervalo entre instruções alto	§5.2 — amarre as leituras às páginas
Escrita não chega ao equipamento	Modo de disparo exige condição que não ocorre	§5.2 — confira a variável ou tecla de disparo
Uma linha da tabela nunca executa	Primeiro byte diferente de 0x5A, ou comprimento zero	§5.1 ou §6.1
Funciona na bancada, falha no painel	Instalação do barramento RS485, ou cabo longo demais para RS232/TTL	§2 — terminação, malha, derivação

Este documento reúne os parâmetros publicados pelo fabricante para as duas implementações de Modbus e a prática de integração acumulada pela AGT Displays. Os valores foram conferidos contra a documentação de cada variante; quando elas divergem, a divergência está declarada em vez de reconciliada. Confirme com o suporte da AGT Displays qual variante se aplica ao seu módulo antes de configurar.