MPPT DC-DC XY7025

YIYIELECTRONIC XY7025.pdf

Admite control anaógico de 0,3-10 V, con comunicación en serie + comunicación 485 integrada (se pueden utilizar simultaneamente.

El RS485 dice que es compatible con el XY-K485X que es el que yo uso, a deducir que el voltaje que utiliza debe ser el mismo y que a su vez el XY-K485X es compatible con TTL 3.3 y 5.0 V.

Para comunicarse con un arduino o esp32, usar un puerto TTL solo dedicado a este equipo, para evitar conflictos de comunicacióny se puede usar la librería ModbusMaster.h, Implementar rutina CRC en Arduino (la librería ModbusMaster lo incluye)

Arduino: Alias cortos pero legibles (`VOLTAJE_SALIDA` vs `VOUT`)
Nextion: Usar nombres descriptivos añadir prefijo XY7025nID ID del dispositivo esclavo (`XY7025n1_LIMITE_Hvalor`)

```
// Leer Amperios-Hora acumulados
uint16_t CONTADOR_AH_LOW = mppt.readRegister(0x0058); // S-OAH_L Parte baja AH
uint16_t CONTADOR_AH_HIGH = mppt.readRegister(0x0059);  // S-OAH_H Parte alta AH
uint16_t CONTADOR_AH;        // CONTADOR_AH = (CONTADOR_AH_HIGH << 16) | CONTADOR_AH_LOW // mAh

// Enviar a Nextion con nombre descriptivo:
void enviarANextion(uint8_t idDispositivo) {
  Serial.print(nombreDispositivo);
  Serial.print("n");
  Serial.print(idDispositivo);
  Serial.print("_LIMITE_TIEMPO_H=");
  Serial.print(LIMITE_TIEMPO_H);
  Serial.write(0xFF); Serial.write(0xFF); Serial.write(0xFF); // Delimitadores Nextion

  Serial.print(nombreDispositivo);
  Serial.print("n");
  Serial.print(idDispositivo);
  Serial.print("_CORRIENTE_FIN_CARGA=");
  Serial.print(CORRIENTE_FIN_CARGA);      // Quizás más comodo corrienteFinCarga (como entero por el nextion)
  Serial.write(0xFF); Serial.write(0xFF); Serial.write(0xFF);
}

// Escribir registro (función 0x06)
void writeRegister(uint16_t reg, uint16_t value) {
  mppt.writeSingleRegister(reg, value);
}
```

Voltaje de salida ajustable de 0 a 70 V, corriente de salida de 0 a 25 A.

I. Breve descripción del acuerdo
El protocolo de comunicación es el protocolo MODBUS-RTU, este producto solo admite códigos de función 0x03, 0x06, 0x10; comunicación

Interfaz para el puerto de serie TTL;

II. Introducción al Acuerdo
El protocolo de comunicación es el protocolo MODBUS-RTU, este producto solo admite códigos de función 0x03, 0x06, 0x10; comunicación
Interfaz para el puerto de serie TTL;
La información se transmite asincrónicamente, y el modo Modbus-RTU se mide en bytes de 11 bits.

Formato de palabra (datos de serie)	Binario de 10 bits
Posición inicial	1er lugar
Bits de datos	8.o lugar
Paridad	ninguno
Posición de parada	1er lugar ¿10.o lugar?

Estructura del marco de datos:

Intervalo de marco de datos	Código de dirección	Código de función	Área de datos	Verificación de CRC
Más de 3.5 bytes	1 byte	1 byte	N bytes	2 bytes

Antes de enviar datos, se requiere que el bus de datos sea tiempo estacionario, es decir, el tiempo de transmisión de datos es mayor que 3.5 (por ejemplo: la tasa de baudios es
5ms en 9600) El envío del mensaje debe comenzar al menos 3.5 bytes de intervalo de pausa, todo el mensaje
El marco debe ser un flujo de transmisión de datos continuo si hay una parada de más de 3.5 bytes antes de que se complete el marco
En un momento, el dispositivo receptor actualizará el mensaje incompleto y asumirá que el siguiente byte es el campo de dirección de un nuevo mensaje.
Del mismo modo, si un nuevo mensaje comienza con el mensaje anterior en menos de 3.5 caracteres, la configuración recibida
Se considerará como una continuación de las noticias anteriores.

1.1 Código de dirección (esclavo):
El código de dirección es el primer byte (8 bits) de cada marco de información de comunicación, de 1 a 255. Este byte indica que el usuario establece que el usuario
El esclavo que establece la dirección recibirá la información enviada por el host. Cada esclavo debe tener un código de dirección único y
Y solo los esclavos que cumplan con el código de dirección pueden responder al mensaje de respuesta. Cuando el esclavo devuelve la información, los datos de retorno son
Comienzan los códigos de dirección respectivos. El código de dirección enviado por el host indica la dirección esclava a la que devuelve el esclavo
El código de dirección indica la dirección esclava del bucleback. El código de dirección correspondiente indica de dónde proviene la información.

1.2 Código de función:
El código de función es el segundo byte transmitido para cada marco de información de comunicación. El código de función que puede definir las regulaciones de comunicación de Modbus es

1 a 127. Envíe como solicitud de host y dígale al esclavo qué acción debe realizarse a través del código de función. Como esclavo
El código de función devuelto por el esclavo es el mismo que el código de función enviado desde el host, e indica que el esclavo ha respondido al host y
Y se han realizado operaciones relacionadas. Esta máquina solo admite códigos de función 0x03, 0x06 y 0x10.

Código de función	Definición	Operación (binario)
0x03	Leer datos de registro	Leer datos de uno o más registros
0x06	Escribe un solo registro	Escriba un conjunto de datos binarios en un solo registro
0x10	Escribe múltiples registros	Escriba múltiples conjuntos de datos binarios a múltiples registros

1.3 Área de datos
El área de datos incluye qué información debe ser devuelta por el esclavo o qué acciones realizar. Esta información puede ser datos (Tales como: conmutación de entrada/salida, entrada/salida analógica, registro, etc.), dirección de referencia, etc.
Por ejemplo, el host le dice al esclavo que devuelva el valor del registro (incluida la dirección de inicio del registro que se lee a través del código de función 03.
y lea la longitud del registro), luego los datos devueltos incluyen la longitud de los datos y el contenido de datos del registro.

0x03 Escribir formato de host de función de lectura

Código de dirección	Código de función	Registre la dirección de inicio	Número de direcciones de registro n (1 ～ 32)	Código de verificación de CRC
1 byte	1 byte	2 byte	2 byte	2 byte

0x03 Función de lectura. Formato de retorno de esclavos

Código de dirección	Código de función	Devolver el número de registros n*2	Registrar datos	Código de verificación de CRC
1 byte	1 byte	1 byte	2*n bytes	2 byte

0x06 Escribir formato de host de función de registro único

Código de dirección	Código de función	Registre la dirección de inicio	Registrar datos	Código de verificación de CRC
1 byte	1 byte	2 byte	2 byte	2 byte

0x06 Escribir Función de registro único. Formato de retorno de esclavos

Código de dirección	Código de función	Registre la dirección de inicio	Registrar datos	Código de verificación de CRC
1 byte	1 byte	2 byte	2 byte	2 byte

0x10 Escribir formato de host de la función de registro múltiple

Código de dirección	Código de función	Registre la dirección de inicio	Registro de la dirección Cantidad N (1 ~ 32)	Número de bytes escritos 2*N	Registrar datos	Código de verificación de CRC
1 byte	1 byte	2 byte	2 byte	1 byte	2*n bytes	2 byte

0x10 Escribir formato de host de esclavos de registro múltiple

Código de dirección	Código de función	Registre la dirección de inicio	Número de direcciones de registro n	Código de verificación de CRC
1 byte	1 byte	2 byte	2 byte	2 byte

Fábrica predeterminada tasa de baudio 115200 dirección del dispositivo 1

Introducción del registro de protocolo (los datos en una sola dirección de registro son datos de doble byte)

Alias XY7025 alias arduino & for Nextion add prefix XY7025nID	Definición	N Bytes	Coma decimal	unidad	Leer/ Escribir	Dirección de registro (decimal)	Dirección de registro (hexadecimal)
V-SET `VOLTAJE_CONFIG`	Config. valores perfil actual de voltaje e.g., 25.98V	2	2	V	R/W	0	0x0000
I-SET `CORRIENTE_CONFIG`	Config. valores perfil actual de corriente 35.986	2	3	A	R/W	1	0x0001
VOUT	Voltaje de salida e.g., 24.76V	2	2	V	R	2	0x0002

VOLTAJE_SALIDA							
IOUT CORRIENTE_SALIDA	Corriente de salida e.g., 10.468A	2	3	A	R	3	0x0003
POWER POTENCIA_SALIDA	Potencia de salida e.g., 240.98W	2	2	W	R	4	0x0004
UIN VOLTAJE_ENTRADA	Voltaje de entrada e.g., 24.39V	2	2	V	R	5	0x0005
AH-LOW ACTUAL_AH_BAJO	AH actuales Bajo 16 bits	2	0	mAh	R	6	0x0006
AH-HIGH ACTUAL_AH_ALTO	AH actuales Alto 16 bits	2	0	mAh	R	7	0x0007
WH-LOW ACTUAL_WH_BAJO	WH actuales Bajo 16 bits	2	0	mWh	R	8	0x0008
WH-HIGH ACTUAL_WH_ALTO	WH actuales Alto 16 bits	2	0	mWh	R	9	0x0009
OUT_H TIEMPO_ACTIVO_H	Tiempo Activo [Horas]	2	0	H	R	10	0x000A
OUT_M TIEMPO_ACTIVO_M	Tiempo Activo [Minutos]	2	0	M	R	11	0x000B
OUT_S TIEMPO_ACTIVO_S	Tiempo Activo [Segundos]	2	0	S	R	12	0x000C
T_IN TEMP_INTERNA	Temperatura interna e.g., 85.2 ºC/ºF	2	1	F/C	R	13	0x000D
T_EX TEMP_EXTERNA	Temperatura externa e.g., 85.2 ºC/ºF	2	1	F/C	R	14	0x000E

LOCK BLOQUEO_TECLADO	Bloqueo de Teclado 0 Desbloqueado 1 Bloqueado	2	0	-	R/W	15	0x000F
PROTECT ESTADO_PROTECCIONES	Estado de protecciones	2	0	-	R/W	16	0x0010
CVCC MODO_CARGA	Modo de Carga 0 Modo CV 1 Modo CC	2	0	-	R	17	0x0011
ONOFF ESTADO_SALIDA	Estado de salida	2	0	-	R/W	18	0x0012
F-C ESCALA_TEMP	Símbolo de temperatura	2	0	-	R/W	19	0x0013
B-LED BRILLO_PANTALLA	Nivel de brillo de la luz de fondo	2	0	-	R/W	20	0x0014
SLEEP TIEMPO_APAGADO_PANT	Tiempo encendido de pantalla	2	0	M	R/W	21	0x0015
MODEL MODELO	Modelo de producto	2	0	-	R	22	0x0016
VERSION VERSION_FIRMWARE	Número de versión de firmware	2	0	-	R	23	0x0017
SLAVE-ADD DIRECCION_ESCLAVO	Dirección del dispositivo	2	0	-	R/W	24	0x0018
BAUDRATE_L BAUDRATE	Tasa de baudios	2	0	-	R/W	25	0x0019
T-IN-OFFSET AJUSTE_TEMP_INT	Corrección de temperatura interna e.g., 85.2 ºC/ºF	2	1	F/C	R/W	26	0x001A
T-EX-OFFSET AJUSTE_TEMP_EXT	Corrección de temperatura externa e.g., 85.2 ºC/ºF	2	1	F/C	R/W	27	0x001B
BUZZER ESTADO_BUZZER	Interruptor de timbre	2	0	-	R/W	28	0x001C

EXTRACT-M MODO_PREDEFINIDO	Perfiles preconfigurados	2	0	-	R/W	29	0x001D
DEVICE ESTADO_DISPOSITIVO	Estado del equipo	2	0	-	R/W	30	0x001E
MPPT-SW HABILITAR_MPPT	Habilitar como MPPT o como fuente conmutada	2	0	-	R/W	31	0x001F
MPPT-K FACTOR_CALIB_MPPT	Factor calibración MPPT	2	0	-	R/W	32	0x0020
BatFul CORRIENTE_FIN_CARGA	**umbral de corriente mínima** para considerar la batería cargada	2	0	-	R/W	33	0x0021
CW-SW HABILITAR_POT_CONST	Activa/desactiva el modo de potencia constante.	2	0	-	R/W	34	0x0022
CW POTENCIA_CONSTANTE	Potencia objetivo	2	0	W	R/W	35	0x0023
V-SET VOLTAJE_CONFIG_PERFIL	Voltaje de salida Perfil rápido e.g., 10.50V	2	2	V	R/W	80	0x0050
I-SET CORRIENTE_CONFIG_PERFIL	Corriente de salida Perfil rápido e.g., 35.986A	2	3	A	R/W	81	0x0051
S-LVP UMBRAL_SUBTENSION	Umbral Protección Subtensión e.g., 10.50V	2	2	V	R/W	82	0x0052
S-OVP UMBRAL_SOBRETENSION	Umbral Protección Sobretensión e.g., 14.60V	2	2	V	R/W	83	0x0053
S-OCP UMBRAL_SOBREC	Umbral Protección Sobrecorriente	2	3	A	R/W	84	0x0054

ORRIENTE	e.g., 35.986A						
S-OPP UMBRAL_SOBREPOTENCIA	Umbral Proteccción Sobrepotencia	2	1	W	R/W	85	0x0055
S-OHP_H LIMITE_TIEMPO_H	Temporizador apagado HORAS	2	0	H	R/W	86	0x0056
S-OHP_M LIMITE_TIEMPO_M	Temporizador apagado MINUTOS	2	0	M	R/W	87	0x0057
S-OAH_L CONTADOR_AH_BAJO	Contador de Amperios-hora entregados (parte baja 16 bits)	2	0	mAh	R/W	88	0x0058
S-OAH_H CONTADOR_AH_ALTO	Contador de Amperios-hora entregados (parte alta 16 bits)	2	0	mAh	R/W	89	0x0059
S-OWH_L CONTADOR_WH_BAJO	Contador de Wattios-hora entregados (parte baja 16 bits)	2	0	10 mWh	R/W	90	0x005A
S-OWH_H CONTADOR_WH_ALTO	Contador de Wattios-hora entregados (parte alta 16 bits)	2	0	10 mWh	R/W	91	0x005B
S-OTP TEMP_MAX_INTERNA	Temperatura máxima interna	2	0	F/C	R/W	92	0x005C
S-INI SALIDA_INICIO	Estado inicial de la salida al energizar el equipo	2	0	-	R/W	93	0x005D
S-ETP HABILITAR_NTC_EXT	Habilita sensor de temperatura (NTC) externo	2	0	-	R/W	94	0x005E

NOTAS:

BAUDRATE_L (Registro 0x0019) Significado del registro de tasa de baudios:
0: 9600 bps
1: 14400 bps
2: 19200 bps
3: 38400 bps
4: 56000 bps
5: 576000 bps
6: 115200 bps
(7: 2400 bps, 8: 4800 bps - soportado por algunos dispositivos)

// Función para cambiar baud rate (usar con precaución!) 115200 Por defecto

```
//   0       1       2       3       4       5       6       7      8
// 9600, 14400, 19200, 38400, 56000, 576000, 115200, 2400, 4800
void configurarBaudRate(uint8_t nuevoBaud) {
  // Validar valor permitido (0-8)
  if(nuevoBaud <= 8) {
    // Escribir nuevo valor en registro 0x0019
    escribirRegistro(0x0019, nuevoBaud);

    // IMPORTANTE: Reiniciar comunicación con nueva velocidad
    delay(100);
    Serial2.updateBaudRate(obtenerVelocidadBaud(nuevoBaud));
  }
}
```

El producto tiene 10 grupos de almacenamiento de perfiles (M0 a M9). Cada grupo contiene 14 registros de datos (direcciones 0x20 a 0x2D en hexadecimal), donde:

- **M0**: Perfil predeterminado que se carga al energizar el equipo
- **M1**-**M2**: Perfiles de acceso rápido (invocados desde el panel frontal)
- **M3**-**M9**: Perfiles de almacenamiento general
  La dirección inicial de cada grupo se calcula como:
  `0x0050 + (NúmeroGrupo * 0x0010)`

Perfil	Fórmula	Dirección Inicial	Rango de Registros
M0	0x0050 + (0*0x10)	**0x0050**	0x0050 - 0x005D
M1	0x0050 + (1*0x10)	**0x0060**	0x0060 - 0x006D
M2	0x0050 + (2*0x10)	**0x0070**	0x0070 - 0x007D
M3	0x0050 + (3*0x10)	**0x0080**	0x0080 - 0x008D
...	...	...	...
M9	0x0050 + (9*0x10)	**0x00E0**	0x00E0 - 0x00ED

Estructura de cada perfil (14 registros):

```
0x00: V-SET    // Voltaje objetivo
0x01: I-SET    // Corriente objetivo
0x02: S-LVP    // Protección subtensión
0x03: S-OVP    // Protección sobretensión
...
0x0D: S-ETP    // Sensor térmico externo

// Leer parámetro específico de un perfil
uint16_t leerParametroPerfil(uint8_t numPerfil, uint8_t offset) {
  uint16_t dirBase = 0x0050 + (numPerfil * 0x0010);
  return leerRegistro(dirBase + offset);
}

// Ejemplo: Obtener voltaje configurado en M3
```

```
uint16_t UMBRAL_SOBREPOTENCIA_M3 = leerParametroPerfil(3, 0x05);  // Lee 0x0085
```

```
// Cargar perfil M2 (acceso rápido)
void cargarPerfilRapido(uint8_t numPerfil) {
  uint16_t dirBase = 0x0050 + (numPerfil * 0x0010);
  escribirRegistro(EXTRACT_M_REG, numPerfil);  // EXTRACT-M = número perfil
}
```

Registro de estado de protección (PROTECT):
Los valores indican el tipo de protección activada:

- **0**: Funcionamiento normal
- **1**: OVP (Protección por sobretensión)
- **2**: OCP (Protección por sobrecorriente)
- **3**: OPP (Protección por sobrepotencia)
- **4**: LVP (Protección por subtensión de entrada)
- **5**: OAH (Protección por capacidad máxima de salida alcanzada)
- **6**: OHP (Protección por tiempo máximo de funcionamiento alcanzado)
- **7**: OTP (Protección por sobretemperatura interna)
- **8**: OEP (Protección por ausencia de salida)
- **9**: OWH (Protección por energía máxima de salida alcanzada)
- **10**: ICP (Protección por corriente de entrada máxima)
- **11**: ETP (Protección por temperatura externa)

Comportamiento del sistema durante protecciones:

1. **Protecciones "blandas"** (códigos 5,6,9):
    - Advertencia sin apagado
    - Sistema sigue operando con limitaciones
2. **Protecciones "duras"** (códigos 1,2,4,7,11):
    - Apagado inmediato de salidas
    - Requiere reset manual o por comando
    - LED de error parpadeante

```
// Dirección registro PROTECT = 0x0010
String decodificarProteccion(uint16_t valor) {
  switch(valor) {
    case 0: return "NORMAL";
    case 1: return "SOBRETENSION";
    case 2: return "SOBRECORRIENTE";
```

```
    case 3: return "SOBREPOTENCIA";
    case 4: return "SUBTENSION";
    case 5: return "LIM_CAPACIDAD_AH";
    case 6: return "LIM_TIEMPO_OPER";
    case 7: return "SOBRETEMP_INTERNA";
    case 8: return "SIN_SALIDA";
    case 9: return "LIM_ENERGIA_WH";
    case 10: return "LIM_CORRIENTE_ENTRADA";
    case 11: return "TEMP_EXTERNA";
    default: return "DESCONOCIDO";
  }
}

// Uso:
uint16_t estado = leerRegistro(0x0010);
Serial.print("Estado protección: ");
Serial.println(decodificarProteccion(estado));
```

BatFul Es el **umbral de corriente mínima** para considerar la batería cargada (típicamente **2-5% de la capacidad**).

“**parte baja/alta 16 bits**” son 32 bits dividido en dos paquetes.

// Ejemplo lectura acumulador AH:
uint16_t low = readRegister(S_OAH_L);
uint16_t high = readRegister(S_OAH_H);
uint32_t total_Ah = (high << 16) | low; // Combina 32 bits

El documento especifica **"10mWh"** → Multiplicar valor leído ×10.
Ejemplo: Si S-OWH_L=500 → Energía real = 500 × 10 = 5000mWh / 1000= 5Wh

Comportamiento en arranque (S-INI):

0: Salida apagada al energizar (requiere activación por comando)
1: Salida automáticamente ON al recibir energía

Para el estado de voltaje/corriente constante (CVCC), los valores de lectura son 0 y 1:

- **0**: Modo CV (voltaje constante)
- **1**: Modo CC (corriente constante)

Este registro (**CVCC** en dirección **0x0011**) indica la fase actual de carga de la batería:

Modo CC, Corriente Constante, Máxima corriente disponible, voltaje aumenta

Modo CV, Voltaje Constante, Voltaje máximo, corriente disminuye

[Batería descargada] → |Inicio carga|
[Modo CC] → |Voltaje alcanza máximo|
[Modo CV] → |Corriente < BatFul|
[Flotación]

BatFul Determina transición CV → Flotación
VOUT En CV debe igualar voltaje configuración
IOUT En CC debe ser cercano a I-SET

Tabla de valores típicos:

Parámetro	Modo CC	Modo CV
Voltaje	< V-SET	= V-SET
Corriente	= I-SET	< I-SET
Eficiencia	85-95%	90-98%

CW-SW

- **Función real:** Activa/desactiva el **modo de potencia constante**.

Traducción profesional:
`HABILITAR_CP` (Constant Power) o `MODO_POTENCIA_CONST`

Explicación:

- `ON (1)`: El regulador mantiene una **potencia de salida fija** (configurada en CW)
- `OFF (0)`: Opera en modo normal (seguimiento MPPT o PWM)

Para CW-SW (HABILITAR_CP):

"Activa el modo de potencia constante. Cuando está habilitado, el regulador mantendrá la potencia de salida en el valor configurado en CW, ignorando el punto de máxima potencia (MPPT)."

Valores:

- `0`: Modo normal (seguimiento MPPT activo)
- `1`: Modo potencia constante (CW)

Para CW (SETPOINT_POTENCIA):

"Potencia objetivo (en vatios) para el modo de operación constante. Rango típico: 10-100% de la capacidad del regulador."

Ejemplo:

- Regulador de 30A @ 12V → Potencia máxima = 360W
- CW seguro = 50-300W (evitar sobrecalentamiento)

Aplicaciones prácticas del modo CP:

Escenario	Configuración	Beneficio
Alimentar equipo crítico	CW=150W	Elimina fluctuaciones por nubes/sombra
Pruebas de estrés	CW=90% capacidad	Verifica estabilidad térmica
Cargas resistivas fijas	CW=Potencia nominal	Evita picos dañinos

Consejos críticos de implementación:

1. **Límites:**
    - CW debe ser < **Capacidad máxima** del regulador (ej: 300W para uno de 360W)
    - **Nunca superar** `Vbat × Imax_regulador`

Para la función de salida de conmutación (ONOFF), los valores de lectura/escritura son 0 y 1:

Este registro (**ONOFF** en dirección **0x0012**) controla el estado del MOSFET de salida del MPPT:

0: Estado APAGADO
1: Estado ENCENDIDO

La salida puede no activarse si hay protecciones activas:

```
void encenderSalidaSegura() {
  if(leerRegistro(0x0010) == 0) {  // PROTECT = 0 (normal)
    controlarSalida(true);
  } else {
    Serial.println("Protección activa - No se puede encender");
  }
}
```

Este registro (B-LED en dirección 0x0014) controla la intensidad de la retroiluminación de la pantalla LCD/LED del MPPT:

El nivel de brillo de la luz de fondo (B-LED) tiene un rango de lectura/escritura de 0-5, donde el nivel 0 es el más oscuro y el nivel 5 es el más brillante.

Este registro (EXTRACT-M en dirección 0x001D) activa los 10 perfiles preconfigurados (M0 a M9) del MPPT:

Para la función de llamada rápida de grupos de datos (EXTRACT-M), los valores de escritura válidos son 0-9. Al escribir un valor, se cargará automáticamente el grupo de datos correspondiente.

Ejemplo de uso con perfiles específicos:

```
void seleccionPerfilAutomatico() {
  // Basado en tipo de batería detectado
  float voltaje = leerVoltajeBateria();

  if(voltaje > 13.0 && voltaje < 14.4) {
    cambiarPerfil(1); // LiFePO4
  } else if(voltaje > 12.0 && voltaje < 13.0) {
    cambiarPerfil(2); // Plomo-ácido
  } else {
    cambiarPerfil(0); // Predeterminado
  }
}
```

Casos de uso avanzados:

Rotación estacional de perfiles:

```
void ajusteEstacional() {
  int mes = obtenerMes();

  if(mes >= 3 && mes <= 9) { // Verano
    cambiarPerfil(3); // Perfil verano
  } else { // Invierno
    cambiarPerfil(4); // Perfil invierno
  }
}
```

Perfiles basados en carga conectada:

```
void perfilSegunCarga() {
  float potencia = leerPotenciaActual();

  if(potencia > 200) { // Carga pesada
    cambiarPerfil(5); // Perfil alta potencia
  } else if(potencia > 50) { // Carga media
    cambiarPerfil(1); // Perfil estándar
  } else { // Carga ligera
    cambiarPerfil(6); // Perfil eficiencia
  }
}
```

Código de verificación de errores (CRC):

Tanto el maestro como el esclavo pueden usar el código de verificación para determinar si la información recibida es correcta. Debido al ruido electrónico u otras interferencias, la información a veces puede corromperse durante la transmisión. El código de verificación de errores (CRC) puede detectar si la información tiene errores durante la transmisión de datos de comunicación, permitiendo descartar datos erróneos (tanto en envío como recepción), lo que aumenta la seguridad y eficiencia del sistema. El CRC (Código de Redundancia Cíclica) del protocolo MODBUS ocupa 2 bytes (16 bits). El CRC es calculado por el dispositivo transmisor (maestro) y se coloca al final de la trama. El dispositivo receptor (esclavo) recalcula el CRC de la información recibida y compara si coincide con el CRC recibido. Si no coinciden, indica un error. El CRC se transmite con el byte menos significativo primero (little-endian).

Algoritmo de cálculo CRC-16:
(1) Inicializar un registro de 16 bits con valor hexadecimal FFFF (todos 1s); este registro se llama registro CRC;
(2) Realizar XOR entre el primer byte de datos (primer byte de la trama) y el byte bajo del registro CRC, guardando el resultado en el registro CRC;
(3) Desplazar el contenido del registro CRC 1 bit a la derecha, rellenando el bit más alto con 0, y verificar el bit desplazado;
(4) Si el bit desplazado es 0: repetir paso 3; si el bit desplazado es 1: realizar XOR entre el registro CRC y el polinomio A001 (1010 0000 0000 0001);
(5) Repetir pasos 3 y 4 hasta 8 desplazamientos, procesando así todo el byte;

(6) Repetir pasos 2 a 5 para cada byte siguiente de la trama;
(7) Después de procesar todos los bytes, intercambiar los bytes alto y bajo del registro CRC;
(8) El contenido final del registro CRC es el código CRC.

La libreria ModbusMaster.h (https://github.com/4-20ma/ModbusMaster),
ya tiene incorporado el código de verificación de errores (CRC) puede detectar si la información tiene errores durante la transmisión de datos de comunicación con este dispositivo.

Implementación práctica (NO necesitas hacer nada del CRC):

Ejemplo de código que YA INCLUYE verificación CRC:

```
#include <ModbusMaster.h>

ModbusMaster nodo;  // Crea un objeto Modbus

void setup() {
  Serial2.begin(115200);  // Tu UART para RS485
  nodo.begin(1, Serial2); // Configura dirección esclavo 1
}

void loop() {
  // ¡La librería maneja el CRC automáticamente!
  uint8_t resultado = nodo.readHoldingRegisters(0x0002, 1); // Leer VOUT

  if (resultado == nodo.ku8MBSuccess) {
    // CRC correcto - Procesar datos
    uint16_t voltaje = nodo.getResponseBuffer(0);
    Serial.print("Voltaje: ");
    Serial.println(voltaje / 10.0, 1); // 2 decimales → ÷10
  }
  else {
    // Error (incluyendo CRC incorrecto)
    Serial.print("Error Modbus: 0x");
    Serial.println(resultado, HEX);
    manejarError(resultado);
  }
  delay(1000);
}
```

Códigos de error relevantes (que incluyen fallos de CRC):

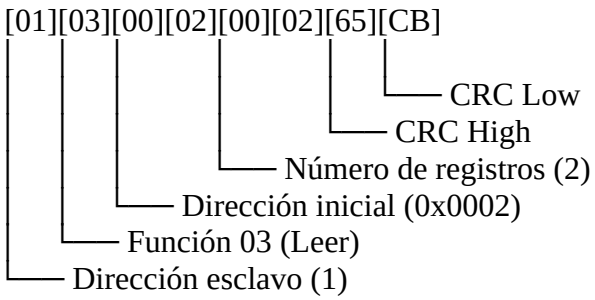
`ku8MBSuccess`	0x00	**Éxito** (CRC correcto y respuesta válida)
`ku8MBInvalidCRC`	0xE2	**CRC inválido** en la respuesta recibida
`ku8MBResponseTimedOut`	0xE1	Timeout (no llegó respuesta o CRC)
`ku8MBIllegalFunction`	0x01	Función no soportada por el esclavo

Ejemplo de Comunicación

Ejemplo 1: Maestro lee valores de voltaje y corriente de salida

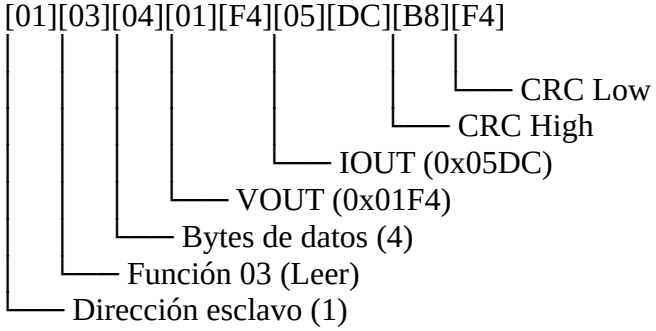
Trama enviada por el Maestro:

Elemento	Bytes	Valor	Descripción
Dirección Esclavo	1	0x01	Enviado al esclavo dirección 1
Código Función	1	0x03	Leer registros
Dirección Inicial	2	0x0002	Dirección del primer registro (VOUT)
Número de Registros	2	0x0002	Leer 2 registros consecutivos
CRC	2	0x65CB	CRC calculado por el maestro



Trama de respuesta del Esclavo (ejemplo con 5.00V y 1.500A):

Elemento	Bytes	Valor	Descripción
Dirección Esclavo	1	0x01	Del esclavo dirección 1
Código Función	1	0x03	Leer registros
Bytes de Datos	1	0x04	4 bytes de datos (2 registros × 2 bytes)
Registro 0x0002 (VOUT)	2	0x01F4	Valor de voltaje de salida
Registro 0x0003 (IOUT)	2	0x05DC	Valor de corriente de salida
CRC	2	0xB8F4	CRC calculado por el esclavo



Implementación con ModbusMaster:

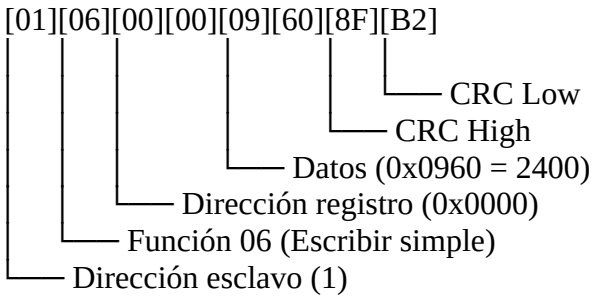
1. Código para leer voltaje y corriente:

```
#include <ModbusMaster.h>
#include <SoftwareSerial.h>

ModbusMaster mppt;

void setup() {
  Serial2.begin(115200);
  mppt.begin(1, Serial2); // Esclavo dirección 1
}

void leerDatosSalida() {
  // Leer 2 registros empezando en 0x0002 (VOUT e IOUT)
  uint8_t resultado = mppt.readHoldingRegisters(0x0002, 2);

  if (resultado == mppt.ku8MBSuccess) {
    // Decodificar valores (recordar coma decimal)
    float voltaje = mppt.getResponseBuffer(0) / 100.0; // 0x01F4 = 500 → 5.00V
    float corriente = mppt.getResponseBuffer(1) / 1000.0; // 0x05DC = 1500 →
1.500A

    Serial.print("Voltaje: ");
    Serial.print(voltaje, 2);
    Serial.println(" V");

    Serial.print("Corriente: ");
    Serial.print(corriente, 3);
    Serial.println(" A");

    Serial.print("Potencia: ");
    Serial.print(voltaje * corriente, 2);
    Serial.println(" W");
  } else {
    Serial.print("Error lectura: 0x");
    Serial.println(resultado, HEX);
  }
}

void loop() {
  leerDatosSalida();
  delay(2000);
}
```

Ejemplo 2: Maestro configura voltaje a 24.00V

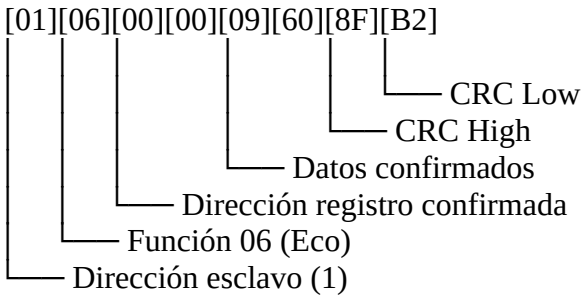
Trama enviada por el Maestro:

Elemento	Bytes	Valor	Descripción
Dirección Esclavo	1	0x01	Para esclavo dirección 1
Código Función	1	0x06	Escribir registro simple
Dirección Registro	2	0x0000	Dirección registro (V-SET)
Datos a Escribir	2	0x0960	Valor de voltaje a configurar
CRC	2	0x8FB2	CRC calculado por el maestro



Trama de respuesta del Esclavo (eco de confirmación):

Elemento	Bytes	Valor	Descripción
Dirección Esclavo	1	0x01	Del esclavo dirección 1
Código Función	1	0x06	Escribir registro simple
Dirección Registro	2	0x0000	Dirección registro confirmada
Datos Escritos	2	0x0960	Valor confirmado
CRC	2	0x8FB2	CRC calculado por el esclavo



Implementación con ModbusMaster:

Código para configurar voltaje:

```
#include <ModbusMaster.h>

ModbusMaster mppt;

void setup() {
```

```
  Serial2.begin(115200);
  mppt.begin(1, Serial2); // Esclavo dirección 1
}

bool configurarVoltaje(float voltaje) {
  // Convertir 24.00V → 2400 (2 decimales)
  uint16_t valorRegistro = voltaje * 100;

  // Escribir en registro 0x0000 (V-SET)
  uint8_t resultado = mppt.writeSingleRegister(0x0000, valorRegistro);

  if (resultado == mppt.ku8MBSuccess) {
    Serial.print("Voltaje configurado a: ");
    Serial.print(voltaje, 2);
    Serial.println(" V");
    return true;
  } else {
    Serial.print("Error configurando voltaje: 0x");
    Serial.println(resultado, HEX);
    return false;
  }
}

void loop() {
  // Ejemplo: Configurar a 24.00V
  if(configurarVoltaje(24.00)) {
    delay(5000);
    // Configurar a 12.00V después de 5 segundos
    configurarVoltaje(12.00);
  }
  delay(10000);
}
```

Análisis del valor 0x0960:

```
uint16_t valorHex = 0x0960;    // = 2400 decimal
float voltaje = valorHex / 100.0; // = 24.00V
```

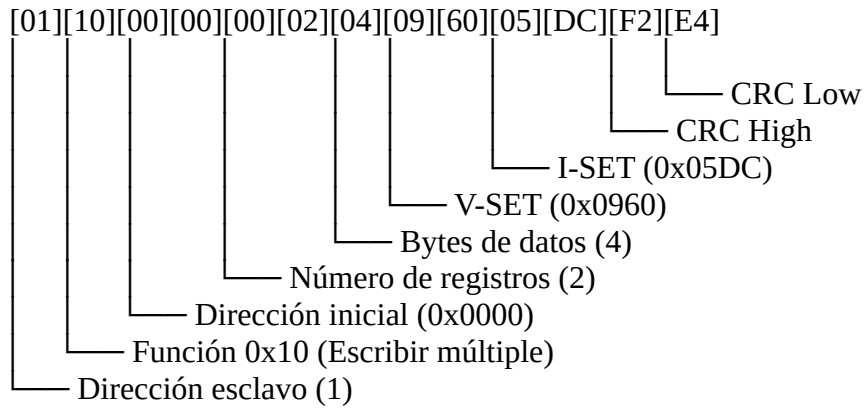
Confirmación: El registro **V-SET** usa **2 decimales** (valor × 100)

Ejemplo 3: Maestro configura voltaje a 24.00V y corriente a 15.00A

Trama enviada por el Maestro:

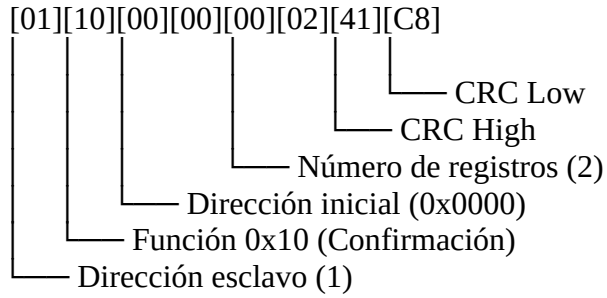
Elemento	Bytes	Valor	Descripción
Dirección Esclavo	1	0x01	Para esclavo dirección 1
Código Función	1	0x10	Escribir múltiples registros
Dirección Inicial	2	0x0000	Dirección inicial (V-SET)

Elemento	Bytes	Valor	Descripción
Número de Registros	2	0x0002	2 registros a escribir
Bytes de Datos	1	0x04	4 bytes de datos (2×2 bytes)
Registro 0x0000 (V-SET)	2	0x0960	Valor de voltaje
Registro 0x0001 (I-SET)	2	0x05DC	Valor de corriente
CRC	2	0xF2E4	CRC calculado por el maestro



Trama de respuesta del Esclavo (confirmación):

Elemento	Bytes	Valor	Descripción
Dirección Esclavo	1	0x01	Del esclavo dirección 1
Código Función	1	0x10	Escribir múltiples registros
Dirección Inicial	2	0x0000	Dirección inicial confirmada
Número de Registros	2	0x0002	Número de registros confirmado
CRC	2	0x41C8	CRC calculado por el esclavo



Implementación con ModbusMaster:

Código para configuración múltiple:

```
#include <ModbusMaster.h>

ModbusMaster mppt;

void setup() {
  Serial2.begin(115200);
  mppt.begin(1, Serial2);
}
```

```
bool configurarVyI(float voltaje, float corriente) {
  // Preparar datos para escritura múltiple
  mppt.setTransmitBuffer(0, voltaje * 100);   // V-SET: 24.00V → 2400
  mppt.setTransmitBuffer(1, corriente * 1000); // I-SET: 15.00A → 15000

  // Escribir 2 registros empezando en 0x0000
  uint8_t resultado = mppt.writeMultipleRegisters(0x0000, 2);

  if (resultado == mppt.ku8MBSuccess) {
    Serial.print("Configurado: ");
    Serial.print(voltaje, 2);
    Serial.print("V, ");
    Serial.print(corriente, 2);
    Serial.println("A");
    return true;
  } else {
    Serial.print("Error configuración múltiple: 0x");
    Serial.println(resultado, HEX);
    return false;
  }
}

void loop() {
  // Configurar 24.00V y 15.00A
  configurarVyI(24.00, 15.00);
  delay(10000);
}
```

Análisis de los valores:

Voltaje (0x0960):

```
uint16_t valorHex = 0x0960;    // = 2400 decimal
float voltaje = valorHex / 100.0; // = 24.00V
```

Corriente (0x05DC):

```
uint16_t valorHex = 0x05DC;    // = 1500 decimal
float corriente = valorHex / 100.0; // = 15.00A
```

Nota: ¡Aquí hay una discrepancia! En el ejemplo 2, corriente usaba 3 decimales (÷1000), pero aquí usa 2 decimales (÷100). Esto necesita verificación.

Protección externa (S-ETP):

Si=1: El regulador monitorea un pin externo para NTC

¿Requiere circuito adicional?:
A[NTC 10k] --> B[Divisor de tensión] --> C[Pin ADC del MPPT]

Registros Duplicados:

`V-SET` e `I-SET` aparecen en 0x0000-0x0001 y 0x0050-0x0051
Muy probablemente los de 0x005X son para configuración de perfil rápido

Pide al vendedor:

Confirmar si los acumulados (S-OAH/S-OWH) se resetean manualmente o por tiempo

DUDAS:

```
// Escribir registro (función 0x06)
void writeRegister(uint16_t reg, uint16_t value) {
  mppt.writeSingleRegister(reg, value);
}
```

COMUNICACION con IA:

continuo con una duda:
0x0000 V-SET R/W 2 decimales
0x0001 I-SET R/W 3 decimales

Registros de Monitoreo
0x0002 VOUT Read 2 decimales
0x0003 IOUT Read 3 decimales

y los valores correspondientes a las M0 – M9

0x0050 V-SET R/W 2 decimales
0x0051 I-SET R/W 2 decimales

está claro que los 0x0002-3 son la lectura actual
y que los 0x0050-1 son los valores objetivos,
pero entonces ya conociendo toda la documentación y
viendo el detalle de la diferencia de 2 a 3 decimales,
¿que podríamos deducir?