LARMIA

ANVÄNDARHANDBOK
EVO MODBUS DRIVARE

ANSVARSBEGRÄNSNING

All information i denna handbok har kontrollerats noggrant och bedöms vara korrekt. Emellertid lämnar Larmia Control AB inga garantier vad gäller manualens innehåll. Användare av denna manual ombeds rapportera felaktigheter, tvetydigheter eller oklarheter till Larmia Control AB, för eventuella korrigeringar i framtida utgåvor. Informationen i denna handbok kan ändras utan föregående meddelanden.

Mjukvaran som beskrivs i handboken levereras under licens från Larmia Control AB och får endast användas eller kopieras enligt licensvillkoren. Ingen del av denna bok får återges eller överföras i någon form eller på något sätt, elektroniskt eller mekaniskt, för något som helst ändamål utan uttryckligt skriftligt medgivande från Larmia Control AB.

COPYRIGHT

VARUMÄRKEN

MS-DOS, Windows, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10 och Windows 11 är registrerade varumärken som tillhör Microsoft Corporation.

Andra produktnamn som förekommer i denna bok används enbart i identifieringssyfte och kan vara ägarens registrerade varumärken.

Januari 2026

Version: 25.12.8.2

Innehållsförteckning

Modbus

Översikt

Modbus definierar fyra olika registertyper:

Register	Funktion
Input Register	Läsbara analoga 16-bitarsregister
Holding Register	Läs- och skrivbara analoga 16-bitarsregister
Input Status	Läsbara digitala bit-register
Coil Status	Läs- och skrivbara digitala bit-register

De olika objekttyperna i Larmias system läser som standard från olika typer av register:

Objekttyp	Register
Analog in	Input Register
Puls	Input Register
Analog ut	Holding Register
Manöver	Coil Status
Indikering	Input Status
Larm	Input Status

Ibland förekommer det att enheter t.ex. inte implementerar Input Register utan endast använder Holding Register, vilket medför att standardinställningen för ett Analog in-objekt inte fungerar. Detta kan dock enkelt lösas via inställningar i drivaren.

> *Se avsnittet **Anpassningar** för mer information.*

JBus

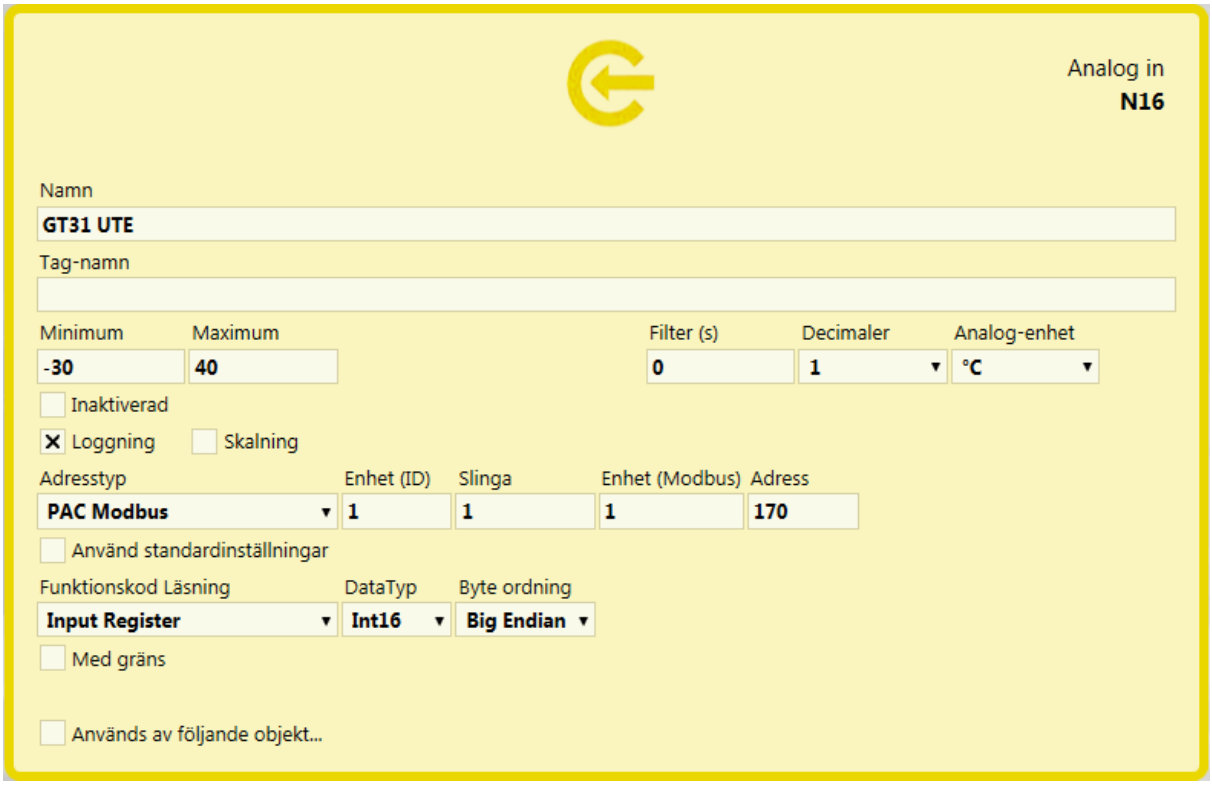
JBus är ett protokoll som baseras på Modbus, och de två protokollen är i stort sett identiska. Den egentliga skillnaden rör registeradresseringen; Modbus-register börjar på adress 0 medan JBus-register börjar på adress 1.

Larmias Modbus-drivare hanterar både Modbus och Jbus.

> ***OBSERVERA** skillnaderna i adressering när du editerar objekten.*

Editering

Använd adresstypen **PAC Modbus** i ED10.



För denna adresstyp finns fyra fält: **Enhet (ID)**, **Slinga**, **Enhet** och **Adress**

Fält	Beskrivning
Enhet(ID)	PAC:ens PLC-id
Slinga	Nummer på slingan
Enhet	Enhetsnummer
Adress	Modbus-adress (registeradress)

> ***OBSERVERA*** *att den första Modbus-adressen är 0. Vissa adresslistor börjar på 1 och man ska då ta adressen enligt listan -1.*
>
> *Om **Använd standardinställningar** avaktiveras visas ett antal objektspecifika fält där man kan ändra t.ex. datatyp.*
>
> *Se **Objektinställningar** för mer information om detta.*

Automatisk styrning av analoga- och digitala utgångar

Om ett värde eller en koppling har angetts i **Auto**-fältet för en analog- eller digital utgång så tar PAC:en över styrningen av dessa objekt.

Om **Auto**-fältet lämnas tomt bestäms värdet av Modbus-enheten.

Ibland vill man kunna avaktivera auto-styrningen. Ex. Under dagtid vill man manuellt i en lokal display kunna ända ett värde i enheten vi kommunicerar med. Efter dagens slut skall värdet automatiskt sättas till ett bestämt värde. Man kan då använda sig utav systemfunktionen **BLOCK**.

Exempel:

Om man i utgången på villkoret skriver ALT(A,BLOCK,22) så kommer objektet se till att modbusregistret får värdet 22 få A är FRÅN. Då A är TILL kommer objektet bara läsa från registret.

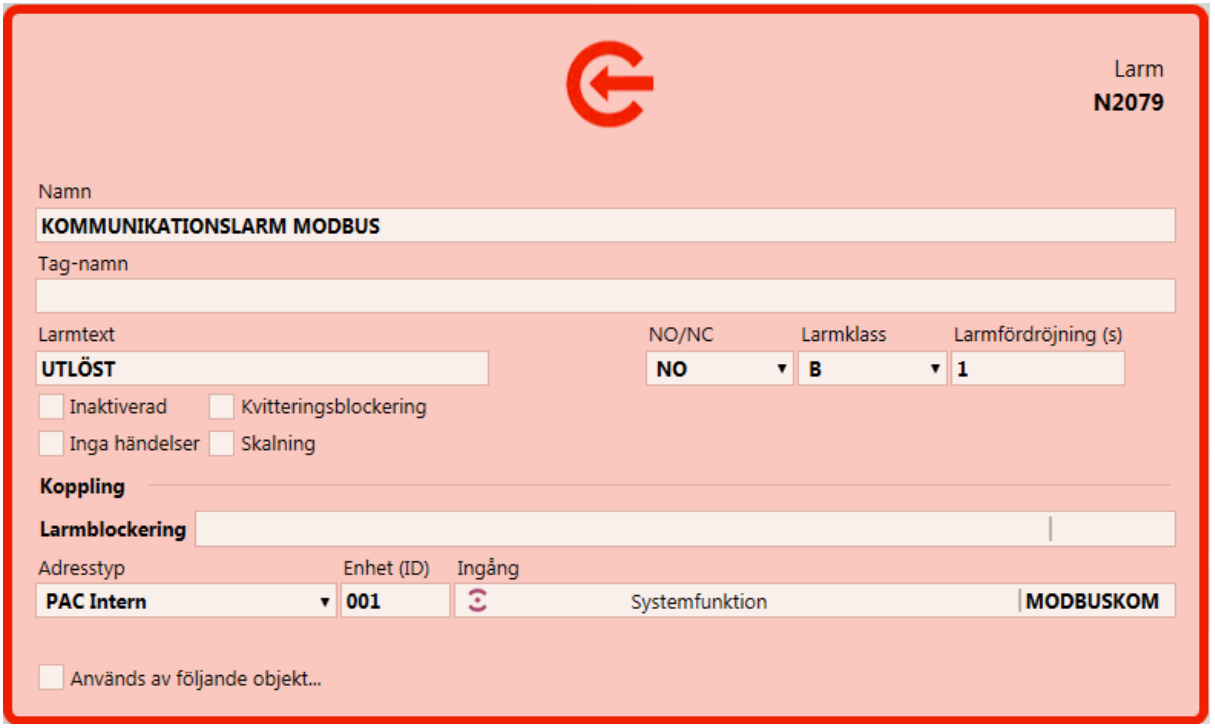
OBSERVERA *Systemfunktion BLOCK finns endast i Evo Enheter. Evo SCADA, Evo Avalon eller PAC från och med paket 19.6.26.0*

Prioriterade objekt

Objekt som är editerade som prioriterade kommer att frågas av mellan varje annan fråga.

Kommunikationskontroll

För kontroll av Modbus-kommunikationen används systemfunktionen **MODBUSKOM**. Denna funktion är **TILL** då något Modbus-objekt indikerar kommunikationsfel.



Objektinställningar

Funktionskoder för skrivning

Modbus definierar fyra funktionskoder för att skriva data till register:

Funktionskod	Beskrivning
Single Coil	Skriver en bit till ett register
Multiple Coils	Skriver flera bitar till ett eller flera register
Single Register	Skriver ett 16-bitarstal till ett register
Multiple Registers	Skriver flera 16-bitarstal till flera register

De olika objekttyperna i Larmias system använder följande funktionskoder vid skrivning:

Objekttyp	Funktionskod
Analog ut	Single Register
Manöver	Single Coil

Datatyper

Ett analogt register (Input- eller Holding Register) består enligt Modbus-standarden av 16 bitar och läses/skrivs som ett 16-bitars heltal med tecken (Int16).

I vissa enheter kan register ibland innehålla en annan datatyp, och för att Modbus-drivaren ska tolka dessa värden på ett korrekt sätt måste motsvarande datatyp väljas för objektet.

Datatyp	Förklaring
Byte	8 bitar med tecken (-128…127)
UByte	8 bitar utan tecken (0…255)
Int16	16 bitar med tecken (-32768…32767)
UInt16	16 bitar utan tecken (0…65535)
Int32	32 bitar med tecken
UInt32	32 bitar utan tecken
Int64	64 bitar med tecken
UInt64	64 bitar utan tecken
Float	32-bitars flyttal enl. IEEE-754
Double	64-bitars flyttal enl. IEEE-754
BIT0-15	Maskar ut en bit ur ett 16-bitarstal

Byteordning

Modbus-standarden anger att protokollet är ett *Big-Endian*-protokoll. Med detta menas att den mest signifikanta delen (byte) av ett 16-bitarstal skickas före den mindre signifikanta.

Vissa enheter använder istället omvänd byteordning (*Little-Endian*), vilket innebär att den minst signifikanta delen skickas först.

Byteordningen har en stor inverkan på hur ett värde tolkas av mottagaren. Exempel:

Det decimala talet 171 motsvarar den hexadecimala bytesekvensen **00 AB**. Om man ändrar byteordningen får man bytesekvensen **AB 00** vilket motsvarar det decimala talet 43776.

Wordswap

Vid läsning av värden som är större än 16 bitar (32/64 bitar, float eller double) kan det mellan olika enheter skilja i vilken ordning som varje 16-bitarsord (word) kommer. Genom att aktivera **Wordswap** vänds ordningen för hur talet tolkas av Modbusdrivaren.

Exempel utan wordswap: `AB CD 01 23`.

Exempel med wordswap: `01 23 AB CD` .

Standardinställningar

I de flesta fall behöver man inte tänka på datatyper, byteordning och wordswap utan man kan använda de standardinställningar som genereras då man skapar objektet.

Om **Använd standardinställningar** är aktiverad döljs dessa inställningar och objektet konfigureras enligt nedanstående tabell:

Objekttyp	Läsning	Skrivning	Datatyp	Byteordning	Wordswap
Analog in	Input Register		Int16	Big Endian	
Puls	Input Register		UInt32	Big Endian	Wordswap
Analog ut	Holding Register	Single Register	Int16	Big Endian	
Manöver	Coil	Single Coil			
Indikering	Input Status				
Larm	Input Status				

Skalning av värden

Analoga värden

I Modbus är det vanligt att flyttal skickas som heltal med hjälp av skalning. Om t.ex. värdet 22.1 ska skickas som heltal måste Modbus-slaven multiplicera värdet med 10 (=221) och sedan dividerar Modbus-mastern värdet med 10 (=22.1).

I fältet **Skalningsuttryck** anges hur värdet från en Modbus-enhet ska skalas.

Exempel:

> ***X/10*** *- Dividerar det inlästa Modbus-värdet med 10.*
>
> ***X*50*** *- Multiplicerar det inlästa Modbus-värdet med 50.*
>
> ***OBSERVERA*** *att man vid användning av skalning måste tänka på hur värdet skall skalas då det läses in. Detta gäller både Analog In och Analog Ut.*

Om objektet är en *Analog Ut* kan värdet skickas ut till ett Modbus-register, och i detta fall kommer skalningen automatiskt inverteras drivaren.

Digitala värden

Om ingångsvärdet inte är digitalt kan man använda ett skalningsuttryck för att definiera när objektet anses vara **TILL** resp. **FRÅN**. I fältet **Skalningsuttryck** anges definitionen.

Exempel:

> ***X>8*** *- Om ingångsvärdet är större än 8 är objektet* ***TILL*** *annars* ***FRÅN****.*

Konfigurering

Via ED10 kan inställningar för drivaren göras om man är ansluten mot den. Markera aktuell slinga och gör önskade inställningar. De flesta fälten har ett standardvärde och behöver inte ändras.

TCP-Kommunikation

Fält	Förklaring	Std.värde
IP-Adress	Enhetens eller gateway'ens IP-adress.	
TCP-Port	Enhetens eller gateway'ens portnummer.	502

Slinga

Fält	Förklaring	Std.värde
Aktiv	**true** om slingan används/skall vara aktiv, annars `false` .	`true`
Beskrivning	Valfri text.	
Fördröjning avfrågning	Tiden som drivaren väntar mellan varje fråga (ms).	100
Gräns för kommunikationsfel	Gränsvärde innan drivaren anser att det är kommunikationsfel mot en enhet (s).	60
ReadTimeOut	Timeout för läsning av meddelande (ms).	500
WriteTimeOut	Timeout för skrivning av meddelande (ms).	500

Enhet

Fält	Förklaring	Std.värde
Aktivera avfrågning	Sätt denna till `false` om inte denna enhet skall avfrågas	`true`
Antal analoga register i en läsning	Anger det maximala antalet analoga register som kan avfrågas i en fråga.	32
Antal digitala register i en läsning	Anger det maximala antalet digitala register som kan avfrågas i en fråga.	64

Dessa fält gäller alla enheter på slingan

Enhet X

Samma fält som ovan men värdena gäller då endast den aktuella enheten på slingan.

Felsökning

Om drivaren är aktiverad kan man från drivarkonfigurationen i ED10 komma till detaljerad visning av drivarens status genom att klicka på informationsknappen till höger om drivaren.

De relevanta menyvalen för felsökning är:

Menyval	Förklaring
Channel Communication	Översikt över kommunikationen för alla enheter
Channel X	Information om slinga **X** - Visar status på uppkopplingen
Channel X Device Y	Information enhet **Y** i slinga X. - Visar status och information för alla frågor (*Request*) och svar (*Response*) - Visar värden på alla objekt

Anpassningar

Enheten använder endast Holding Register

Ibland förekommer det att enheter t.ex. inte implementerar Input Register utan endast använder Holding Register, vilket medför att standardinställningen för *Analog In*-objekt inte fungerar.

Detta löses genom att avaktivera **Använd standardinställningar** och välja **Holding Register** i fältet **Funktionskod Läsning** för det aktuella objektet.

Enheter som inte kan skicka tillräckligt stora meddelanden

Som standard är det maximala antal register som skickas i ett meddelande 32st, men vissa enheter klarar inte att skicka så stora meddelanden.

Detta löses genom att ange ett lägre värde i fältet **Antal analoga register i en läsning** eller **Antal digitala register i en läsning** i drivarkonfigurationen.

> *Se **Konfigurering - Enhet** för mer information.*

Hantering av tal som är större än 16 bitar

I fältet **Datatyp** kan man välja om man vill läsa och tolka registervärdet som en annan datatyp, t.ex. 32/64-bitars heltal eller flyttal. Om objektet är en *Analog Ut* kommer funktionskoden **Multiple Registers** automatiskt att användas.

Läs bitar från ett analogt register

En Indikering eller Larmobjekt kan läsa sin status från ett analogt register (Input Register eller Holding Register). Avaktivera **Använd standardinställningar**, välj den faktiska registertypen i **Funktionskod Läsning** och välj slutligen vilken bit som skall läsas i **Datatyp** (*BIT0 - BIT15*).

Läs/skriv flera bitar från ett binärt register in till/från ett analogt objekt

Om man vill hantera flera Coil-register samtidigt i samma läsning och skrivning kan ett Analog Ut-objekt skapas för detta. Avaktivera sedan **Använd standardinställningar**, ange

Coil Status i **Funktionskod Läsning** och *Multiple Coils* i **Funktionskod skrivning**.