Technische Information

SPEEDWIRE DEVICE DISCOVERY

1 Hinweise zu diesem Dokument

1.1 Gültigkeitsbereich

Dieses Dokument gilt für alle SMA Produkte mit SMA Speedwire oder SMA Speedwire/Webconnect.

Kein weiterführender Support verfügbar

Bitte beachten sie, dass wir Ihnen außer den hier verfügbaren Informationen leider keinen weiterführenden Entwicklungssupport für Speedwire Device Discovery anbieten können.

1.2 Zielgruppe

Die in diesem Dokument beschriebenen Tätigkeiten dürfen nur Fachkräfte durchführen. Die Fachkräfte müssen über folgende Qualifikationen verfügen:

- Kenntnisse über IP-basierte Netzwerkprotokolle
- Ausbildung für Installation und Konfiguration von IT-Systemen
- Kenntnisse mit ethernetbasierenden Feldbussen
- Kenntnis und Beachtung dieses Dokuments mit allen Sicherheitshinweisen

2 Speedwire

2.1 Speedwire allgemein

Speedwire ist ein kabelgebundener, ethernetbasierter Feldbus zur Realisierung von leistungsstarken Kommunikationsnetzen in dezentralen PV-Anlagen.

Speedwire verwendet den international etablierten Ethernetstandard, das darauf aufsetzende IP-Protokoll sowie das für PV-Anlagen optimierte Kommunikationsprotokoll SMA Data2+. Dies ermöglicht eine durchgängige 100/1000 Mbit/s Datenübertragung bis zum Wechselrichter sowie eine zuverlässige Überwachung, Steuerung und Regelung der Anlage.

Der Aufbau des Speedwire-Netzwerks kann wahlweise mit Linien-, Stern- oder Baumtopologien realisiert werden (siehe Technische Information „SMA Speedwire Feldbus“ unter www.SMA-Solar.com).

2.2 Speedwire Device Discovery

Beim Speedwire Device Discovery werden die IPv4-Adressen aller im lokalen Netzwerk befindlichen SMA Produkte abgefragt. Jedes mit SMA Speedwire ausgerüstete Produkt kann mittels Speedwire Device Discovery Abfrage im lokalen Netzwerk gefunden werden.

SMA Produkte mit Speedwire erhalten folgendermaßen ihre IPv4-Adresse:

- Durch DHCP von einem im Netzwerk befindlichen Router
- Durch IPv4LL
- Durch eine manuell fest eingestellte Konfiguration

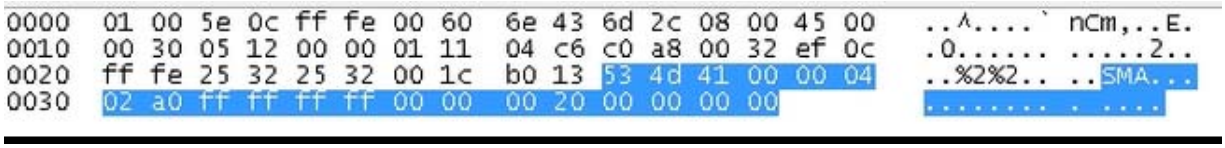
SMA Produkte sind werkseitig auf die IPv4-Adressierung per DHCP eingestellt. Dadurch kann sich die IPv4-Adresse je nach verwendetem Router im laufenden Betrieb verändern. Die Speedwire Device Discovery Abfrage macht es möglich, die IPv4-Adressen der SMA Produkte zu ermitteln. Ist die IPv4-Adresse bekannt und die Modbus®-Kommunikation des SMA Produkts freigeschaltet, können alle weiteren Daten des Produkts per Modbus® ermittelt werden (siehe Kapitel 4 „Weiterführende Informationen“, Seite 6).

Der Speedwire Feldbus ist so aufgebaut, dass der Speedwire Device Discovery Dienst und weitere Basisdienste als IP-/UDP-Telegramme den Port 9522 nutzen.

- Port 9522 ist bei der IANA (Internet Assigned Numbers Authority) für SMA Speedwire Kommunikation eingetragen
- Es werden unter anderem Multicast-Telegramme verwendet

3 Discovery Abfrage und Discovery Antwort

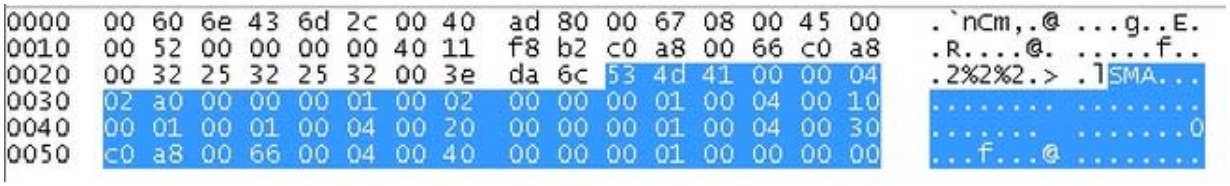
Um SMA Produkte innerhalb eines lokalen Netzwerks aufzufordern sich zu melden, muss folgendes UDP-Datagramm an die Multicast-Adresse 239.12.255.254 über Port 9522 versendet werden (blau markierter Teil):



Alle SMA Produkte, die folgende Voraussetzungen erfüllen, werden ein UDP-Datagramm zurücksenden:

- ☐ Eine Speedwire-Schnittstelle muss vorhanden sein
- ☐ Das abgefragten Produkt muss sich im selben Netzwerk befinden wie das Endgerät mit dem die Abfrage versendet wird
- ☐ Das Produkt muss vom ausgesendeten Multicast erreicht werden können

Das UDP-Datagramm der antwortenden SMA Produkte ist wie in folgender Abbildung aufgebaut (blau markierter Teil):



Aus der Anwort kann die IP-Adresse der SMA Produkte abgeleitet werden. Um Fehler auszuschließen, wird empfohlen die Antwort auszuwerten.

Vorgehen:

- UDP-Datagramm bis einschließlich dem 18. Byte auswerten (innerhalb des blau markierten Teils).

  ☑ Liegt das Sub-Array „534d4100000402A00000000100020000000 1“ als Antwort vor, handelt es sich um ein SMA Produkt.

  Die IPv4-Adressen von SMA Produkten erkennt man anhand der Paket-Absenderadressen (siehe Kapitel 2.1 „Codebeispiel in Java“, Seite 3 und Kapitel 2.2 „Codebeispiel in Python“, Seite 4).

i Richtigen Ethernet-Adapter wählen

Die Antwort der SMA Produkte auf die Speedwire Device Discovery Abfrage kann auf die Multicast-Adresse 239.12.255.254 oder die Adresse des Ethernet-Adapters erfolgen.

Bei der Verwendung mehrerer Ethernet-Adapter darauf achten, dass die Abfrage an den richtigen Ethernet-Adapter gerichtet wird.

In Problemfällen wenden Sie sich an Ihren Netzwerkadministrator.

3.1 Codebeispiel in Java

Das folgende Beispiel zeigt die Verwendung der Speedwire Device Discovery Abfrage in Java. Der aussendende Ethernet-Adapter ist hart codiert.

```
import java.io.*;
import java.net.*;
import java.util.Arrays;

public class SPWDisco {
  /**
   * client to demonstrate the usage of UDP multicast sockets.
   * @throws IOException
 * @throws InterruptedException
   */
  public void multicast() throws IOException, InterruptedException {
    try {
      InetAddress multicastAddress = InetAddress.getByName("239.12.255.254"); // MC Address
      InetAddress adpAdr           = InetAddress.getByName("192.168.0.50");   // host adapter
      final int port = 9522;                                                  // standard port for SPW
      MulticastSocket socket = new MulticastSocket(port);

      socket.setInterface(adpAdr);
      socket.setReuseAddress(true);
      socket.setSoTimeout(5000);
      socket.joinGroup(multicastAddress);

      // send discovery command
      byte[] txbuf = javax.xml.bind.DatatypeConverter.parseHexBinary("534d4100000402a0ffffffff0000002000000000");

      DatagramPacket hi = new DatagramPacket(txbuf, txbuf.length, multicastAddress, port);
      socket.send(hi);
      System.out.println("SPW discover sent\n");


      do {
          byte[] rxbuf = new byte[8192];
          DatagramPacket packet = new DatagramPacket(rxbuf, rxbuf.length);
          socket.receive(packet);
          scanPacket(packet, adpAdr);
        } while (true);                 // should leave loop by SocketTimeoutException
      } catch (SocketTimeoutException e) {
        System.out.println("\n5 sec w/o receiving answers - Timeout, terminating\n");
      }
    }

  private void scanPacket(DatagramPacket packet, InetAddress hostAdr) throws IOException {
      boolean found = true;
      // expected answer:
      byte[] refArr   = javax.xml.bind.DatatypeConverter.parseHexBinary  ("534d4100000402A000000001000200000001");
      byte[] buf      = packet.getData();
      for( int index = 0; index < refArr.length; ++index ) {
           if( buf[index] != refArr[index]) {found = false; }
      }
      if (found==true) {
         InetAddress addr = packet.getAddress();
         if ( Arrays.equals( addr.getAddress(), hostAdr.getAddress())) {
             System.out.println("Response from host Adapter: " + addr);
         }
         else
         {
            System.out.println("Response from Device: " + addr);
         }
      }
  }

      /**
       * MAIN
       */
      public static void main(String[] args) throws Exception {
          SPWDisco client = new SPWDisco();
        client.multicast();
      }
    }
```

3.2 Codebeispiel in Python

Das folgende Beispiel zeigt die Verwendung der Speedwire Device Discovery Abfrage in Python. Der aussendende Ethernet-Adapter ist hart codiert.

```
#
# Example to perform a speedwire discovery using python
#
from twisted.internet.protocol import DatagramProtocol
from twisted.internet import reactor
from twisted.application.internet import MulticastServer

# Change this address to the address of your local network interface
localInterface = '10.2.3.222'

# Nothing to change below this line
spwMCastAdr = '239.12.255.255'
spwPort = 9522

discoveryRequest  = '534d4100000402a0ffffffff0000002000000000'
discoveryResponse = '534d4100000402a000000001000200000001'

class MulticastClientUDP(DatagramProtocol):

   def startProtocol(self):
      print "Joining speedwire multicast group."
      self.transport.joinGroup(spwMCastAdr)
      self.transport.setOutgoingInterface(localInterface)

      print "Sending discovery request."
      data = discoveryRequest.decode('hex')
      self.transport.write(data, (spwMCastAdr, spwPort))

   def datagramReceived(self, datagram, (srcAddress, port)):
      data = datagram.encode('hex')
      if (data.startswith(discoveryResponse)):
         print "Found device: " + srcAddress

def stopReactor():
    print "Discovery finished."
    reactor.stop()

reactor.listenMulticast(spwPort, MulticastClientUDP(), listenMultiple=True)
reactor.callLater(10, stopReactor)
reactor.run()
```

4 Weiterführende Informationen

SMA Dokumente

Links zu weiterführenden Informationen finden Sie unter www.SMA-Solar.com:

Dokumententitel	Dokumentenart
SMA Modbus®-Schnittstelle	Technische Information
SunSpec® Modbus®-Schnittstelle	Technische Information
SMA Speedwire Feldbus	Technische Information

Weitere Dokumente

Dokumententitel	Quelle
Service Name and Transport Protocol Port Number Registry	http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml
Modbus® Application Protocol Specification	http://www.modbus.org/specs.php
Modbus® Messaging Implementation Guide	http://www.modbus.org/specs.php