Marcus Tangermann

Profinet in 96 kByte

Profinet haftet der Ruf an, umfangreiche Ressourcen in Beschlag zu nehmen, was die nachträgliche Integration in Embedded-Systeme problematisch macht. – Zu Unrecht, wie folgender Artikel zeigt.

Halle 6 Stand 420

Der Einsatz von Industrial Ethernet geht vom Trend zur alltäglichen Realität über. Unter einer Vielzahl von Lösungen kristallisieren sich dabei einige Protokolle heraus, die sich bereits auf eine breite Nutzerbasis stützen können. Eines davon ist Profinet, welches den Anspruch erhebt, alle Bereiche der Automatisierung bis hin zu harten Motion-Control-Anforderungen zu erfüllen. Macht dieser Anspruch Profinet zu einer aufgeblähten All-In-One-Lösung, die sich in kostensensitiven Systemen nicht einsetzen lässt? Um diese Frage zu klären, gilt es zunächst einen Blick auf die Technologie und Funktionsweise dieser Industrial-Ethernet-Lösung zu werfen.

Spricht man von Profinet, so ist damit meist die Anwendung für dezentrale Peripherie – Profinet IO – gemeint. Das Komponentenmodell Profinet CBA (Component based Automation) hat bisher kaum Verbreitung gefunden, weshalb im Folgenden die Bezeichnungen Profinet und Profinet IO gleichbedeutend verwendet werden.

Bei dem Protokoll handelt es um eine Standard-Ethernet-Kommunikation gemäß IEEE 802.X (eventuell geltende Einschränkungen werden im Zusammenhang mit der Variante IRT – Isochronous Real Time – besprochen). Als Kommunikationsmodell findet Master-Slave Verwendung. Das bedeutet: Ein Master (der IO-Controller) parametriert verschiedene Slaves (IO-Devices), mit denen er kommuniziert beziehungsweise die er kontrolliert. Daneben erlaubt die Spezifikation so genannte IO-Supervisoren zur Überwachung des Netzwerkes. Geräte werden in der Regel nicht über Adressen angesprochen, sondern über symbolische Namen. Die Zuweisung der Namen und die Zuordnung entsprechender Adressen zu den Namen erfolgt nach der Projektierung während des Hochlaufs der einzelnen Geräte.

Profinet bietet eine automatische Erkennung aller an einem Netzwerk angeschlossener Knoten mit Hilfe des Link Layer Discovery Protocols. Dabei sendet jedes LLDP-fähige Gerät in regelmäßigen Abständen an allen vorhandenen Ethernet-Ports Information über seine Identität. Ebenso empfängt es diese Informationen von anderen angeschlossenen Geräten. Fragt eine Projektierungssoftware diese Informationen von jedem Gerät im Netz ab, so entsteht automatisch die Topologie des Netzwerks. Die Abfrage der Information geschieht in der Regel mit Hilfe des Simple Network Management Protocols (SNMP). Dabei ist es nicht zwangsläufig notwendig, dass ein SNMP-Agent auf jedem Endgerät vorhanden sein muss.

Die Echtzeit-Klassen

Profinet erlaubt die Kommunikation von Anwendungsdaten entweder direkt über Standard-Ethernet-Frames oder über UDP/IP, je nach benötigten Echtzeit-Anforderungen. Um diese Anforderungen zu klassifizieren, hat die Profibus-Nutzerorganisation vier verschiedene Realtime-Klassen definiert:

RT_Class_1:
Innerhalb eines Subnetzes erfolgt eine unsynchronisierte RT-Kommunikation auf Basis von Ethernet-Frames. Diese Kommunikation ist zwingend in ein Gerät zu implementieren. Daten sind innerhalb eines vom IO-Controller vorgegebenen Zeitrasters zu senden.

RT_Class_2:
Geräte, die RT_Class_2 unterstützen, erlauben die synchronisierte oder unsynchronisierte RT-Kommunikation über Ethernet-Frames. Bei der synchronisierten Kommunikation legt der IO-Controller den Beginn des Buszyklus für die IO-Devices fest. Dies führt zu einem vorhersagbaren Worst-Case-Szenario für die Verzögerung der Übertragung von Frames auf dem Weg vom IO-Device zum IO-Controller. Die Synchronisation der Zyklen wird über das Precision Clock Transport Protocol (PTCP) realisiert.

RT_Class_3:
Diese Klasse erweitert die synchrone Datenübertragung, indem es das Scheduling der Sendezeiten für alle Teilnehmer im Netz festlegt. Jeder Teilnehmer im Netzwerk bekommt dazu eine definierte Sendezeit vom IO-Controller zugewiesen. Zusätzlich stellen alle Vermittlungsknoten im Netz (Switches) sicher, dass zum Zeitpunkt der Übertragung die jeweiligen Übertragungswege frei sind. Durch die Umgehung des CDMA-Verfahrens von Ethernet wird eine praktisch verzögerungsfreie Übertragung von IO-Daten im Netzwerk erreicht. Diese Klasse stellt besondere Anforderungen an die Hardware und ist nicht mit Standard-Ethernet-Controllern realisierbar.

RT_Class_UDP:
Hierbei werden Daten zwischen verschiedenen Subnetzen auf Basis von UDP-Datenpaketen übertragen.

Auch wenn die klassische Anwendung von Ethernet eher auf Sterntopologien abzielt, wird in der Automatisierungstechnik eher eine Linienstruktur bevorzugt, da diese untschiedliche Vorteile bei der Verkabelung bietet. Hierfür stellen verschiedene Industrial-Ethernet-Geräte einen integrierten 3-Port-Switch (In, Out, Geräte-Anschluss) zur Verfügung. Für eine erhöhte Ausfallsicherheit implementieren diese Geräte meist zusätzlich das Media Redundancy Protocol (MRP).

Profinet im Netzteil

Die beschriebenen Features machen Profinet zwar zu einer leistungsfähigen Kommunikationslösung, erfordern in Summe aber auch umfangreiche Ressourcen für deren Implementierung. Allerdings: Lässt man die definierten Echtzeit- und Conformance-Klassen von Profinet Revue passieren (s. Kästen auf *S.114* und *S.117*), so wird klar, dass nicht in allen Geräten alle erwähnten Features implementiert sein

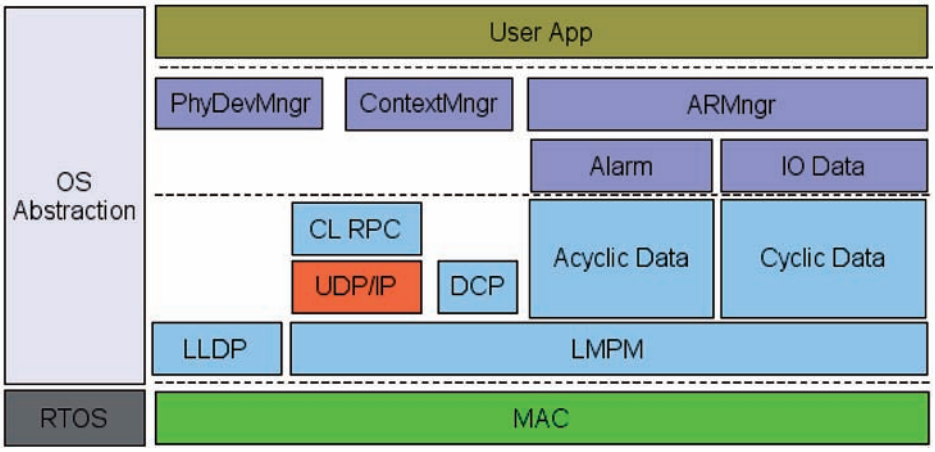
müssen. Vielmehr besteht die Option, je nach Anwendungsfall nur die jeweils benötigten Features einer Conformance Class auszuwählen und zu implementieren, ohne die Spezifikation zu verletzen. Um die konkrete Anwendung zur verdeutlichen, soll als Beispiel die Integration von Industrial Ethernet in ein Netzteil dienen. Erst durch die kontinuierliche Überwachung und Steuerung der Energieversorgung in Automatisierungsgeräten können Kosten effektiv gesenkt werden. Hierzu engagiert sich die PNO im Übrigen auch im Rahmen von ProfiEnergy (Anm. d. Red: siehe hierzu Beitrag in Computer&AUTOMATION 2009, H. 4, S. 36ff.).

Zur Realisierung einer Profinet-Kommunikation genügt in der Regel eine 1-Port-Ethernet-Anschaltung gemäß CC-A. Damit sind lediglich die Standard-Kommunikationsdienste zu implementieren sowie eine einfache Nachbarschaftskennung zu integrieren. Außerdem kann hierfür eine CPU mit Standard-Ethernetcontroller zum Einsatz kommen. Aus diesen einfachen Überlegungen heraus ergeben sich bereits erste Optimierungsmöglichkeiten bezüglich des Funktionsumfangs eines angepassten Profinet-Stacks, ohne dass auch nur eine Zeile Quellcode zu ändern ist. Allerdings genügen diese noch nicht, um den Stack mit nur 96 kByte betreiben zu können. So sind darüber hinaus beim Design der Profinet-Implementierung einige Punkte und mögliche Fallstricke zu beachten.

Die grundlegendste Design-Entscheidung für den zukünftigen Profinet-Stack beschäftigt sich mit dem offensichtlichen Problem geringer Speicherressourcen auf Embedded-Systemen. Insbesondere im Low-Cost-Sektor ist Speicher als Kostenfaktor stets knapp bemessen und zumeist direkt in ein SoC (System on Chip) integriert. Hinzu kommt, das diverse kleinere Prozessoren nicht über eine Memory Management Unit (MMU) verfügen. Die Hauptaufgabe einer MMU besteht in der Abbildung physikalischen Speichers in einem virtuellen, fortlaufenden Adressbereich. Nebeneffekt hiervon ist, dass die Fragmentierung des Speichers gemindert wird. Eine solche MMU steht also in der Regel für die Zielplattform nicht zur Verfügung. Ergo sollten Speicherstrukturen für den zu entwickelnden Stack so weit wie möglich statisch erzeugt werden. Fortlaufendes dynamisches Allokieren – zum Beispiel bei jedem Eintreffen von Ethernet-Frames auf dem IO-Device – würde letztlich zu einer Fragmentierung des Speichers führen, wodurch in kurzer Zeit kein ausreichender Speicher mehr zur Verfügung stünde.

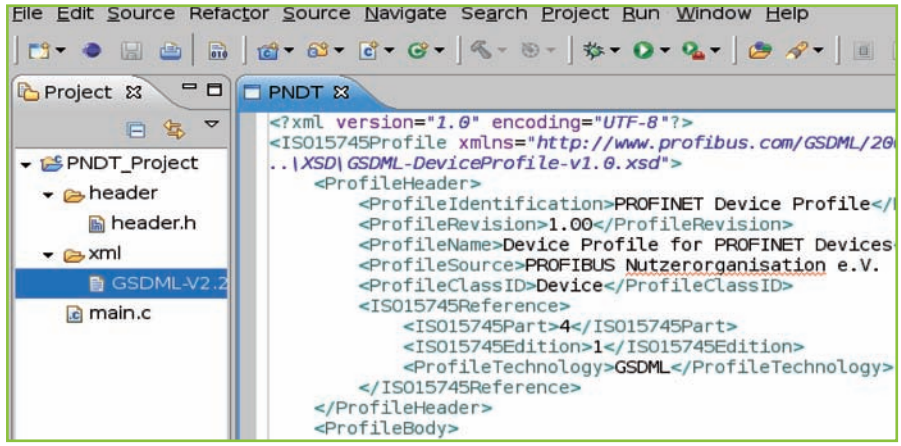
Statische Strukturen haben den weiteren Vorteil, dass sie sich relativ einfach mit Konfigurationstools erzeugen und verwalten lassen. Somit kann sich der Entwickler auf die Implementierung konzentrieren – die Konfiguration des Stacks und der Rumpfapplikation übernimmt das Konfigurationswerkzeug.

Bei der Design-Entscheidung, welche Strukturen zwingend erforderlich sind, gilt die erste Überlegung der Application Relationship (AR) – sprich den Applikationsbeziehungen). Die Spezifika-



(Bilder: Computer & AUTOMATION; Quelle: Port, Messe Nürnberg)

Aufbau der Profinet-Protokollstacks – durch Modularität bleibt die Kommunikation schlank und effizient.



Für die komfortable Auswahl benötigter Features sowie die Generierung entsprechender Funktionsrümpfe für den Profinet-Stack gibt es geeignete Design-Tools, die GSDML, Variablendefinition, Stack-Konfiguration und Anwendungssupport konsistent und fehlerfrei erzeugen.

tion sieht hier im Wesentlichen zwei Möglichkeiten vor:

- Controller AR (IOC-AR): Übertragung zyklischer Ein-/Ausgangsdaten zwischen IO-Controller und IO-Device sowie
- Supervisor AR (IOS-AR): Datenaustausch zwischen IO-Supervisor und IO-Device.

Somit genügt es in der Regel, Datenstrukturen für diese zwei ARs statisch vorzuhalten. Die Daten hierfür sind überschaubar; es ist lediglich darauf zu achten, dass der Speicherbereich für einige Strings (Initiator Name, AR Identifier etc.) ausreichend groß entsprechend der Spezifikation gewählt wird.

Ein weiterer interessanter Punkt bezüglich Speicherverbrauch ist die optimierte Implementierung der Remote Procedure Calls (RPC). Profinet verwendet eine RPC-Kommunikation entsprechend der Spezifikation DCE 1.1 der Open Group (www.opengroup.org). Diese Kommunikation wird während des Kommunikationsaufbaus zwischen IO-Controller/Supervisor und dem IO-Device sowie zum expliziten Lesen und Schreiben von Attributen auf dem IO-Device verwendet. Jeder RPC-Kommunikation wird dabei eine „Session“ zugeordnet. Da pro AR in der Regel nur eine RPC-Session aufgebaut wird, genügt auch hier das Vorhalten von Daten für zwei Sessions.

Eine bestimmte Eigenschaft von RPC kann sich allerdings in Bezug auf flüchtigen Speicher als problematisch erweisen – und zwar die Fragmentierung. Übersteigt die Größe eines RCP-Aufrufs die Nutzdaten-Größe eines Ethernet-Frames (1500 Byte), so kann dieser vom Sender über mehrere Ethernet-Frames verteilt und vom Empfänger wieder vor der Verarbeitung zusammengefügt werden. Eine korrekte Implementierung muss also den Datenpuffer für einen maximal großen RPC-Aufruf bereit halten. Doch wie groß kann ein solcher RCP-Aufruf werden? Betrachtet man sich die Spezifikation näher, so scheint dieser bei Profinet maximal auf 4 kByte anzuwachsen – und zwar für das Lesen von Diagnose-Daten. Die Spezifikation fordert hier die Möglichkeit, einen entsprechend großen Puffer über Profinet auslesen zu können. Daraus würden mindestens 8 kByte pro RPC-Session (4 kByte ausgehender RPC-Daten, 4 kByte eingehende RPC-Daten) resultieren. Ein RCP-Aufruf größer als 1500 Bytes während des Verbindungsaufbaus ist unwahrscheinlich.

Doch auch diesbezüglich gibt es noch weitere Optimierungsmöglichkeiten. Anstatt bei ausgehenden Daten den RCP-Aufruf erst vollständig im Speicher zusammenzusetzen und dann zu fragmentieren (in einzelne Ethernet-Frames aufzuteilen), kann eine „intelligente“ Stack-Implementierung die in den RCP-Aufruf zu integrierenden Daten nach und nach von der Applikation abfragen. Dies reduziert den Datenpuf-

fer für ausgehende RPC-Nachrichten von 4 kByte auf 1500 Byte für einen einzelnen Ethernet-Frame.

Für die Implementierung der zyklischen IO-Daten sollte das Grundprinzip eines Zero-Copy-Stacks gelten. Dabei wird bei der Verarbeitung von Daten weitgehend auf das Kopieren von Datenpuffern verzichtet. Erhält der Stack IO-Daten, die zu versenden sind, so werden diese direkt in einen statischen Puffer übertragen, der das zu versendende Frame enthält. Damit lässt sich der Puffer für Ausgangsdaten auf 1500 Byte reduzieren.

Auch IO-Daten, die ein IO-Device empfängt, kopiert der Stack in einen Empfangspuffer. Bei der weiteren Verarbeitung der Daten wird lediglich per Pointer auf den Puffer zugegriffen; sonstige Kopieraktionen finden nicht statt. Eine Ausnahme stellt die Verarbeitung von RCP-Daten dar, wie dies bereis beschrieben wurde. Somit lässt sich auch hier der Empfangspuffer auf 1500 Byte reduzieren. Last but not least ist ein modularer Aufbau des Stacks zu wählen. Das heißt: Alle nicht zwingend von der IEC-Spezifikation geforderten Elemente sollten per Designtool beziehungsweise Compiler-Optionen deaktivierbar sein.

Zusammenfassend lässt sich festhalten: Die Profinet-Spezifikation bietet bei sorgfältiger Betrachtung viele Optimierungsmöglichkeiten, von denen in diesem Beitrag nur einige beschrieben wurden. Diese stützen sich auf Erkenntnisse, die die Firma Port während der Entwicklung eines eigenen Profinet-IO-Stacks speziell für kleine Embedded-Systeme gesammelt hat. Zur Entwicklungsunterstützung hat man sich hier für die Integration eines Design-Tools auf Eclipse-Basis entschieden, welches die komfortable Auswahl benötigter Features sowie die Generierung entsprechender Funktionsrümpfe für den Profinet-Stack erlaubt. Außerdem wird dabei automatisch eine entsprechende Geräte-Datei (GSD) generiert und mit den jeweiligen Änderungen an der Applikation synchronisiert. *gh*

Marcus Tangermann

ist Geschäftsführer Technologie der Firma Port, Halle/Saale.

Die Conformance Classes

Profinet definiert folgenden Conformance Classes:

CC-A:
Einfaches Gerät mit Unterstützung für Profinet-IO-Basiskommunikation (Alarme, Diagnose, zyklische RT-Kommunikation, Namensauflösung etc.) und einfache Nachbarschaftserkennung.

CC-B:
Wie CC-A, zusätzlich werden Medienredundanz und SNMP sowie Funktionen zum Gerätetausch unterstützt. Die Realisierung der Medienredundanz erfolgt mit Hilfe eines MRP-Clients.

CC-C:
Zusätzlich Unterstützung für RT_Class_3 sowie umfangreiche Redundanz-Unterstützung.