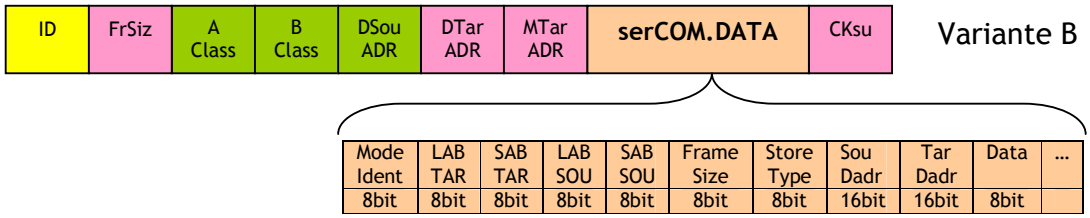
Communication Protocol

1.) Basiseigenschaften

Der Datenaustausch zwischen mehreren Netzwerk-Teilnehmern erfordert eine klare Zugriffsvereinbarung und einheitliche Protokollstrukturen. Widmen wir uns zuerst der Protokollstruktur und deren Aufbau. Grundlegend differenzieren wir innerhalb eines kompletten Datenframes zwischen Kommunikations-, und Datenblock. Der Kommunikationsblock enthält außer der erforderlichen Adressierung u.a. auch eine eindeutige Instruktionsvariable welche den Inhalt des Datenframes und dessen Weiterverarbeitung charakterisiert. Die nachfolgende Skizze verdeutlicht den Aufbau:

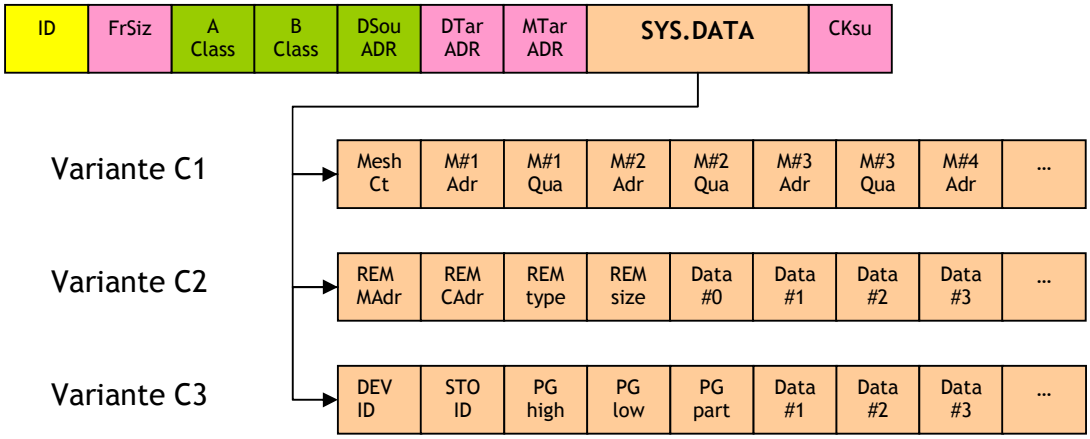
Variante A

Mit einer Länge von *8 Bytes* und dem fehlenden Datenblock sind die Einsatzmöglichkeiten sehr eingeschränkt. In der Tat wird dieser Frame lediglich als Antwort auf vorangegangenen Datenaustausch verschickt – des weiteren als Acknowledge-Frame bezeichnet.
Wie bereits erwähnt liegt den Protokollen eine einheitliche Struktur zu Grunde die eine eindeutige Aufbereitung und Weiterverarbeitung der Daten ermöglicht. Betrachten wir deshalb eine etwas komplexere Variante:



Variante B

Vergleicht man die diese Variante B mit A so fällt sofort der zusätzliche Datenblock (serCOM.DATA) auf. Außer dem üblichen, *8 Bytes* langen Kommunikationsblock ist hier ein zusätzlicher Block variabler Größe integriert, der im Aufbau dem serCOM – Protokoll entspricht und letztendlich den Datenaustausch zwischen den Applikationslayern ermöglicht.

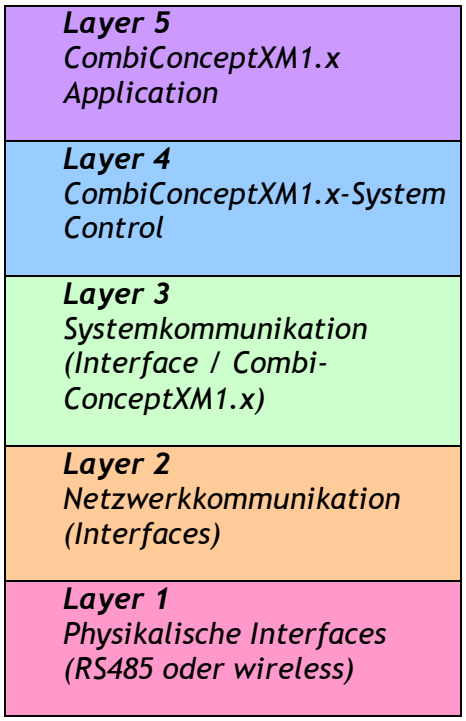


Die Struktur mit den vielfältigsten Spielarten ist zweifelsohne die Variante C. Sie erlaubt die Integration unterschiedlichster Informationsinhalte, wie z.B.: Programmierdaten, Host-Tables,

Remote-Daten uvm. Wie auch bei der Variante B ist der Datenteil in den Grundframe eingebettet. Die Identifikation der jeweiligen Variante, C1,2 oder 3 ist im ID-Byte hinterlegt.

2.) Kommunikationsschichten

Die variierende Komplexität von Systemen setzt angepasste Kommunikationsmechanismen voraus. Der praktischen Umsetzung liegt eine schichtweise Betrachtung zu Grunde:



Layer 1
Der physikalische Layer bestimmt u.a. über den drahtlosen oder drahtgebundenen Datenaustausch zwischen den aktiven Netzwerkteilnehmern. Weitere Parameter, wie: Tranferrate, Sendeleistung, Sende-, Empfangsfrequenz etc. werden ebenfalls bestimmt.

Layer 2
Die Kombination aus Steuercontroller und rfm12-Transceiver (des Weiteren auch als Interface bezeichnet) erlaubt die ständige, vom CombiConceptXM1.x-System unabhängige Überwachung der Netzwerkqualität. Dies beinhaltet die qualitative, sowie quantitative Erfassung aller aktiven Teilnehmer (des Weiteren als Hosts bezeichnet) durch Austausch von speziellen Messages (Scan,Alive,Kill). Hierbei wird zwischen direkt erreichbaren (Direct-) Hosts und entfernt ansprechbaren (Mesh-) Hosts differenziert. Sinngemäß können Meshed-Hosts nur indirekt über einen erreichbaren Direct-Host kontaktiert werden. Bei genauerer Betrachtung der Thematik handelt es sich bei einem Meshed-Hosts um einen, für den direkten Zugriff zu weit entfernten Teilnehmer des Netzwerkes. Welcher Host nun als „Direct“ oder „Meshed“ deklariert wird, bestimmt jedes Interface im Verlauf der Netzwerksuche (Scan-Mode) selbsttätig. Das Ergebnis beinhaltet die verfügbaren Adressen inkl. Qualitätsfaktor (0..99%), wird im Anschluß in die

„scan host-table“- gesichert und steht künftig dem dynamischen Datenaustausch zur Verfügung. Außer o.g. Systemfunktionen übernimmt Layer 2 die Integration des serCOM-Datenpaketes (aus Layer 3) , bzw. Konvertierung des rfm12-Datenpaketes (nach Layer 3).

Layer 3
Die Kommunikation zwischen Interface und CombiConceptXM1.x findet auf Basis der SPI-Interfaces beider Systeme statt. Layer 3 regelt Zugriffsvereinbarungen und beinhaltet Sende- und Empfangspufferspeicher zur Vermeidung von Datenverlust.

Layer 4
Bestimmt den Datenverkehr zwischen Local-, Sub-Area-Bus und Layer 5. Kommunikation mit Sub-Area-Bus

Layer 5
Ereinisabhängige Auswertung und Anforderung von Daten aus untergeordneten Layern.

3.) Interface Setup

Durch das im Mode-Identifier gesetzte Flag: <setup> wird dem Interface der Setup-Mode signalisiert.

Format:

Mode Ident	LAB TAR	SAB TAR	LAB SOU	SAB SOU	Frame Size	Store Type
8bit	8bit	8bit	8bit	8bit	8bit	8bit
Setup=1	dummy	Dummy	dummy	dummy	7	X

Der Wert des Type-Parameters entspricht der gewünschten Setup-Funktion:

0 = invalid
Rückgabewert: -

1 = scan mode
Rückgabewert: -

2 = set new host adress, followed by a kill, alive & a scan message
Rückgabewert: -

3 = rf-transceiver off
Rückgabewert: -

4 = rf-transceiver on
Rückgabewert: -

5 = soft-uart (rs485) off
Rückgabewert: -

6 = soft-uart (rs485) on
Rückgabewert: -

7 = alive message
Rückgabewert: -

8 = kill message
Rückgabewert: -

9 = load host identifier (provides available host-adresses & names)
Rückgabewert:*<mi><4xAdressbytes><framesize><type><16bit Sadr><16bit Tadr><host-adr><6xNamebytes>*

4.) Transfer Protokolle

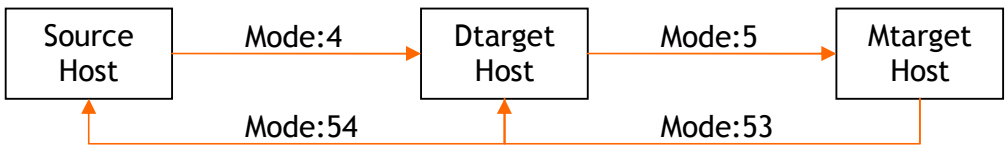
Die Kommunikation zwischen den jeweiligen Interfaces wird auf Basis von rfm12-Protokollen abgewickelt. Grundsätzlich ist zwischen Setup-, und Transfer-Protokollen zu unterscheiden. Während Setup-Protokolle den qualitativen und quantitativen Netzwerkzustand bestimmen, werden Transferprotokolle für den Transport von serCOM-Frames benutzt. Die gewünschte Kommunikationsform ist im [rfm12-instr] Parameter festgelegt:

A) primary:
1 = scan (requests:50)
2 = embedded data to direct host (requests:52)
4 = embedded data to a meshed host - first step receive by the direct host (transmit in order:5)
5 = embedded data from the direct receiving to the meshed-host (transmit in order:53)
6 = embedded data to all available direct hosts (transmit in order:52)
7 = alive message from a host (transmit in order:51)
8 = kill message from a host (transmit in order:51)
9 = lha change message (transmit in order:51)
10 = remote message (transmit in order:55)
11 = RESET IF&SYS (bootloaders will start...)
12 = programming data from the programming interface (transmit in order: 56)
13 = ONLY for PROGRAMMING Interfaces: page programmed, load a new one... (transmit in order: 56)

NOTE: storage segment for primary instructions is always the <RFMtxbuffer>

B)Secondary:
50 = available direct-host (incl.quality) to the calling host (in order of a received:1)
51 = simple acknowledge to the calling host (in order of a received:7/8)
52 = simple acknowledge to the calling host (in order of a received:2/6)
53 = mesh-acknowledge to the calling direct-host (in order of a received:5/transmit:54)
54 = mesh-acknowledge from the direct received to the calling host (in order of a transmitted:4/received:53)
55 = remote acknowledge (in order of a transmitted:10)
56 = programming acknowledge (in order of a transmitted:13)

Beispiel Meshed Transfer:

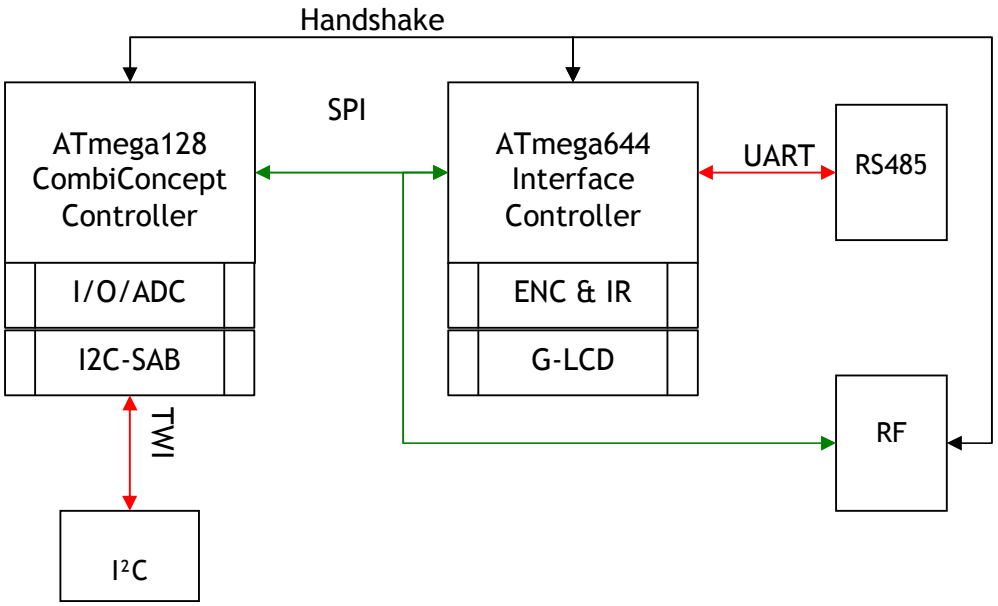


5.) Zugriffsvereinbarungen

Der Netzwerkzugriff wird weitestgehend durch das Interface abgewickelt. Dies bedeutet das CokmbiConceptXM1.:
⇨ erwartet keine Acknowledges nach Datenversand
⇨ generiert keine Acknowledges nach Datenempfang

Acknowledges werden folglich, lediglich auf Interfaceebene ausgetauscht !
Bedingt durch diese Neuerungen wurden allen relevanten Softwarebausteinen des CombiConceptXM1.x-Systems:
⇨ die Acknowledgegenerierung und Auswertung entnommen
⇨ die damit verknüpften Warteschleifen entnommen
Zur Vermeidung von Datenverlust bzw. Kollisionen wurde dem CombiConceptXM1.x, sowie dem Interface ein Hardware-Handshaking integriert. Hierfür verbinden außer dem SPI-Port zwei weitere Handshake-Leitungen die beiden Systeme.

6.) Hardwareschema



Wie der CombiConceptXM1.x Dokumentation zu entnehmen ist werden durch das Basissystem:
- ein S(ub)-A(rea)-B(us), der den I2C- Spezifikation entspricht
- und ein L(ocal)-A(rea)-B(us), der den serCOM- Spezifikationen entspricht,

unterstützt. Der SAB ist direkt am Basisystem zugänglich – der LAB über den separaten LAB-Controller (Interface).