Fachschule für Elektrotechnik, Maschinenbautechnik und Metallbautechnik der Landeshauptstadt München	
Klasse :	Name : MUSTERLÖSUNG

Technikerprüfung 2011

Automatisierungstechnik

Zeit : 150 Minuten

	Punkte:	Note :	Unterschrift:
Erstkorrektur			
Zweitkorrektur			

Teil 1 , ohne Unterlagen

Name, Klasse :

1.1 Welche Aussagen sind richtig ?

- O Dynamische Redundanz ist kann im TMR-Betrieb betrieben werden
- O Durch vorbeugende Wartung kann die Burn-In Phase vermieden werden
- O MTBF ist die Abkürzung für „More Time between Failure“ und bedeutet Wartungsbetrieb
- **X** Profinet CBA benutzt IP-Adressen
- O Layer 1 von ProfibusDP ist identisch mit der Spezifikation RS232

1.2 Worin besteht der wesentliche Unterschied in der Hardware von Profibus DP und Profinet ?

- Profibus DP läuft auf RS485 (seriell Zweidraht)

- Profinet läuft auf Ethernet 100baseT

1.3 Ist die Kommunikation in Profinet IRT deterministisch? Warum ?

Ja, weil die Spezialhardware von IRT dafür sorgt, daß mit Hilfe von einer gemeinsamen Zeitbasis im Netz synchron kommuniziert werden kann.

1.4 Bei loser Kopplung ist in manchen Fällen Produktkennzeichnung nötig.

Bei welchen Fertigungsmethoden ist das nötig und warum ?

Wenn in loser Kopplung Losgröße 1 gefahren wird, also lauter individuelle Produkte ohne festen Anlagentakt hergestellt werden.

1.5 In welchen Fertigungsverfahren ist der Begriff „just in sequence“ relevant ?

Bei Losgröße 1 – Produktion (allgemein : bei flexibler Fertigung).

Zulieferer liefert Teile in benötigter Reihenfolge für Fertigung.

1.6 Welche Aussagen sind richtig ?

- **X** Frames in Profibus DP sind etwa 250 Byte lang
- O Frames in Profibus DP sind etwa 1500 Byte lang
- O Frames in Profinet sind etwa 250 Byte lang
- **X** Frames in Profinet sind etwa 1500 Byte lang
- O Frames in Ethernet sind etwa 250 Byte lang
- **X** Frames in Ethernet sind etwa 1500 Byte lang

1.7 Welche Aussagen sind richtig ?

- **X** In Profibus DP ist keine direkte Master-Master Kommunikation möglich
- O Profibus DP ist auch ohne Master funktionsfähig
- O In Ethernet mit TCP/IP kann nur ein Master („Server“) betrieben werden
- O Profinet benötigt zur Signalübertragung nur eine zweipolige Twisted-Pair Leitung

1.8 Beschreiben Sie das Prinzip des „Handshake“-Verfahrens, und nennen Sie Vorteile :

Alle Signale (meist Timing) werden durch Quittungssignale bestätigt.

Vorteile :

- schneller, weil nicht T(min) gewartet werden muß
- sicherer, weil Partnerfunktion geprüft wird

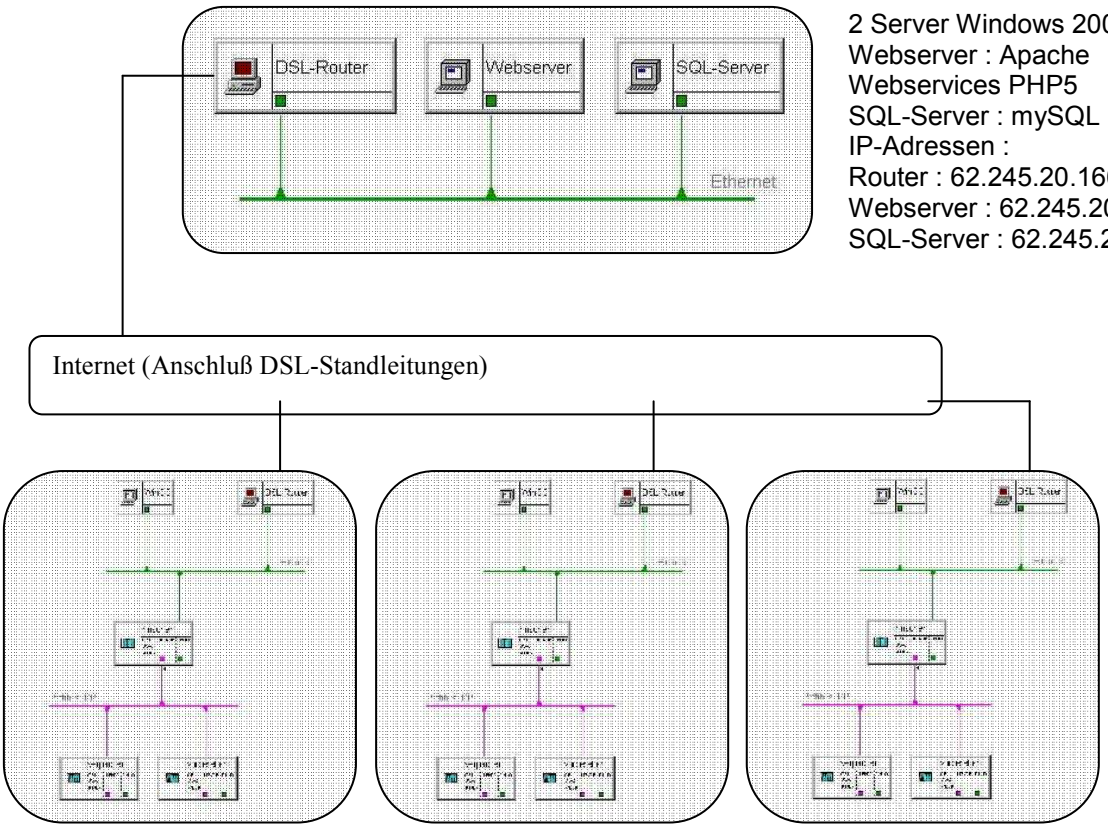
Anlagendokumentation , gesamte Anlagenstruktur :

Wir betrachten die Automatisierungsstruktur der Firma „Müsli24.de“.
Hier können Kunden online selber Müslimischungen zusammenstellen, die dann in der nächstgelegenen Verteilstation vollautomatisch hergestellt, verpackt und verschickt werden.

Angeschlossen an eine zentrale Auftragsannahme mit Webserver und Datenbank bearbeiten europaweit mehrere gleich aufgebaute Verteilstationen die Kundenaufträge. Die Kommunikation zwischen den Stationen erfolgt über das Internet mittels Webservices (XML) :

<u>Zentrale Auftragsannahme :</u>

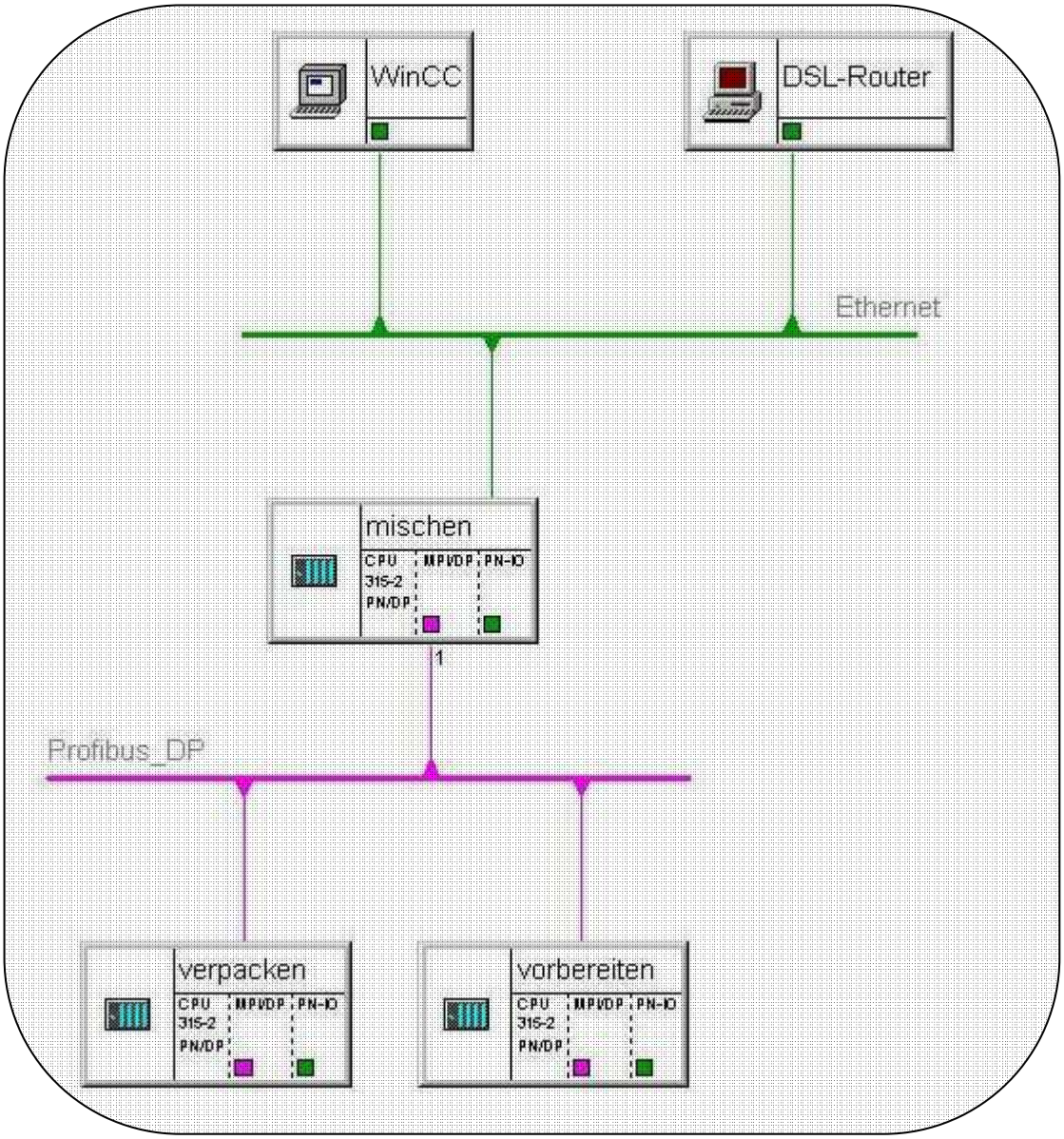
2 Server Windows 2008
Webserver : Apache
Webservices PHP5
SQL-Server : mySQL
IP-Adressen :
Router : 62.245.20.166
Webserver : 62.245.20.167
SQL-Server : 62.245.20.168



<u>Verteilstationen :</u>

Je 1 Server Windows 2008 , Webservices VBScript,WinCC V6.03 SPS Simatic 315-2 dp/pn ,Profibus DP , IP-Adressen Station 3:
Router extern : 194.245.249.12, Router intern : 192.168.178.1
WinCC : 192.168.178.2, SPS-mischen : 192.168.178.3

Anlagendokumentation , Aufbau der Verteilstationen :



Die Master-SPS erfüllt (anders als bei unserer Laboranlage !) neben der Verkettung der Funktionen am Profibus auch selbst eine mechanische Funktion :

Ihr Handshake-FB, der mit dem WinCC-Rechner und dem Profibus kommuniziert, startet mit dem Signal ACTION den Mechanik-FB. Dieser steuert die mechanische Aktion. Wenn die mechanische Aktion beendet ist, setzt der Mechanik-FB das Signal ACTION wieder auf Null.

Anlagendokumentation , Kommunikation :

Zwischen dem WinCC-Rechner und der Master-SPS werden folgende Signale ausgetauscht :

START_Mischen, READY_Mischen

Zwischen der Master-SPS und den beiden Slaves (**Roh**material und **Verp**acken) werden folgende Signale ausgetauscht :

START_Roh, READY_Roh und FEHLER_Roh

sowie : START_Ver, READY_Ver und FEHLER_Ver

Natürlich existieren auch Aufttragsinformationen, in dieser Prüfung werden aber ausschließlich Timingsignale betrachtet !

Damit kann an allen Schnittstellen ein Handshake durchgeführt werden, der nachfolgendem Protokoll genügt :

- Die SPS setzt (beim Einschalten) **READY**=1.
- Wenn ein **START** (START=1) erfolgt, wird READY=0 gesetzt.
  Dies dient als Handshakesignal für START, welches mit erkanntem READY=0 zurückgesetzt wird.
- READY bleibt 0 bis die Funktion beendet und das Gerät damit wieder startbereit ist. Dann wird READY=1 und es kann ein neuer START (= nächster Auftrag) erfolgen.
- In **FEHLER** kann eine Slave- SPS mögliche Betriebsstörungen mit FEHLER=1 zurückmelden.

(in der MISCHEN-Station geht Ready erst wieder auf 1, wenn der gesamte Vorgang - also alle SPS-Aktionen der Prozessebene – beendet ist)

Teil 2, mit Unterlagen

Prozessebene :

Der WinCC-Rechner startet mit Handshake (s.o.) den Vorgang in der Master-SPS. Diese führt den Mischvorgang für ein Produkt aus (Start mit Action=1), und startet nach Beenden des Vorgangs (Action=0) die beiden Stationen Rohmaterial und Verpacken (wieder Handshakes). Neue Gundstoffe werden rohereitet, das eben gemischte Produkt wird versandt.

=> serielle Kopplung von **Mischen** zu **Rohmaterial / Verpacken**

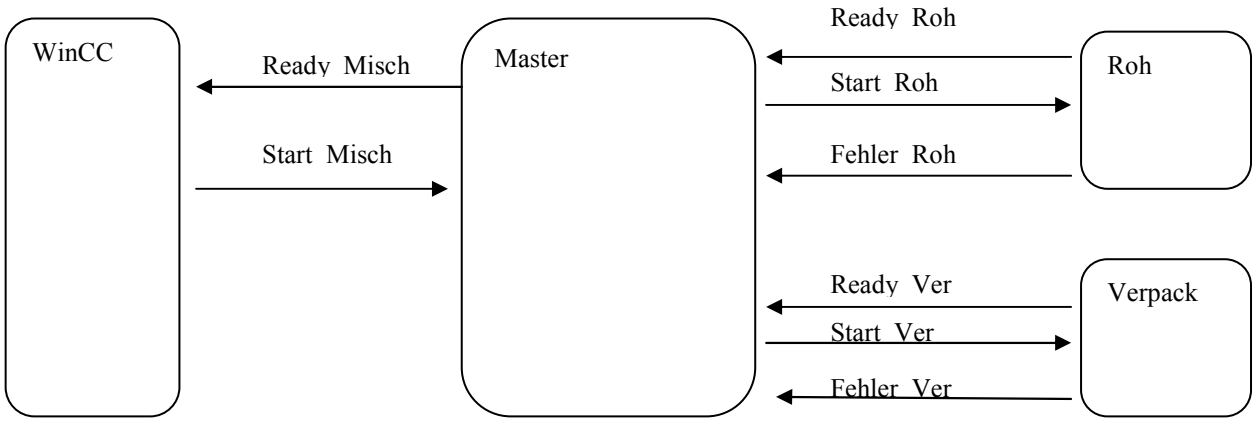
=> parallele, starre Kopplung von **Rohmaterial** und **Verpacken**

Innerhalb der Master-Station Mischen ist die mechanische Aktion durch ein Signal **Action** zu starten. Wenn **Action** wieder Null wird, ist die Mechanik fertig.

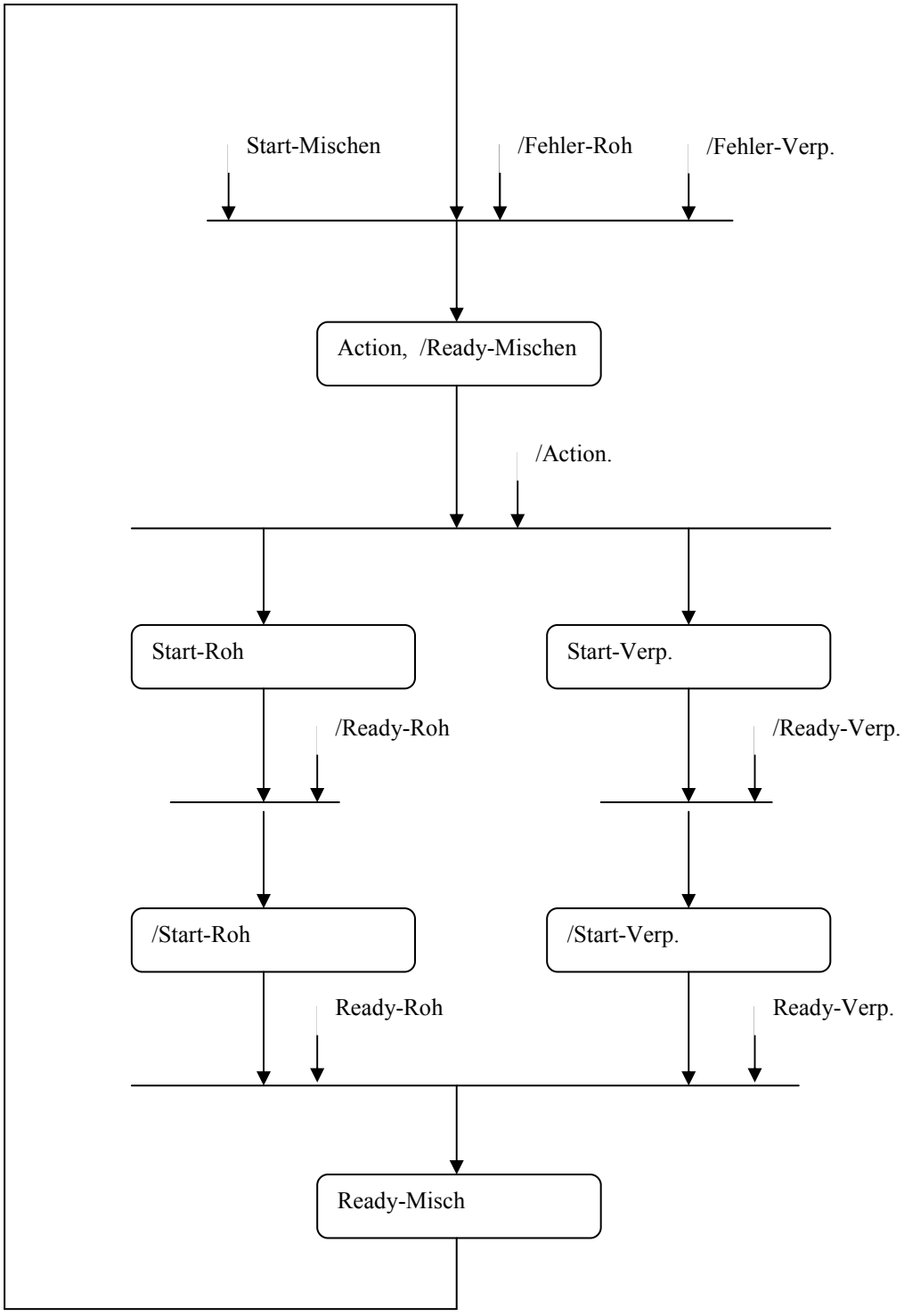
Sollte in einer Slavestation **Fehler** gemeldet werden, läuft der Prozess bis zum nächsten Mischvorgang weiter und stoppt dann, bevor dieser ausgeführt wird.

<u>Aufgaben :</u>

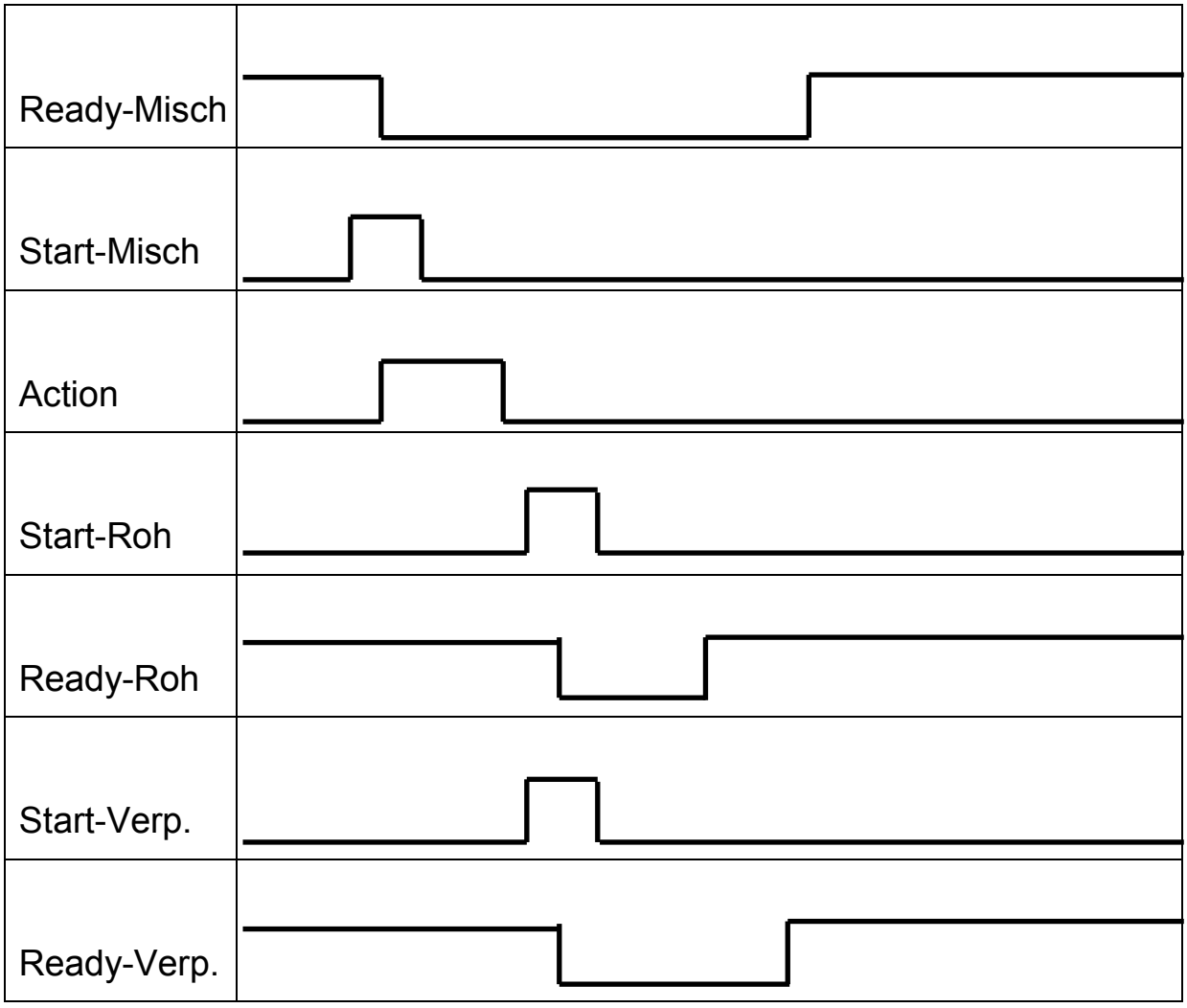
1. Zeichnen Sie ein Blockbild, das alle Signale zwischen WinCC, Master-SPS und den beiden Slaves mit Signalrichtung zeigt . (Action kommt hier nicht vor !)



2. Zeichnen Sie ein Petrinetz für die Mischen-Station, das diesen Ablauf mit der beschriebenen Handshakekommunikation (in beiden Richtungen, also sowohl zu WinCC als auch zu den beiden Slaves) realisiert, und auch die Mechaniksteuerung mit Action beinhaltet.



3. Zeichnen Sie ein Timingdiagramm für die Signale an der Mischen-Station, das diesen Ablauf zeigt. Hierbei lassen Sie zur Vereinfachung die Fehler-Signale weg, das Signal Action (obwohl kein Bussignal) zeichnen Sie aber bitte mit ein.



MES-Ebene :

Wegen sehr hoher Bestellraten soll versucht werden, die Leistung der Anlage durch lose Kopplung zu erhöhen.

Versuchsweise werden dazu alle Steuerungen aus der Prozessebene direkt über Ethernet von WinCC angesteuert.

In WinCC werden folgende Tags definiert :

START_roh, ACK_roh, READY_roh
START _misch, ACK _misch, READY _misch
START _ver, ACK _ver, READY_ver

Mit diesen Signalen wird zur jeweiligen SPS folgender Handshake ausgeführt :

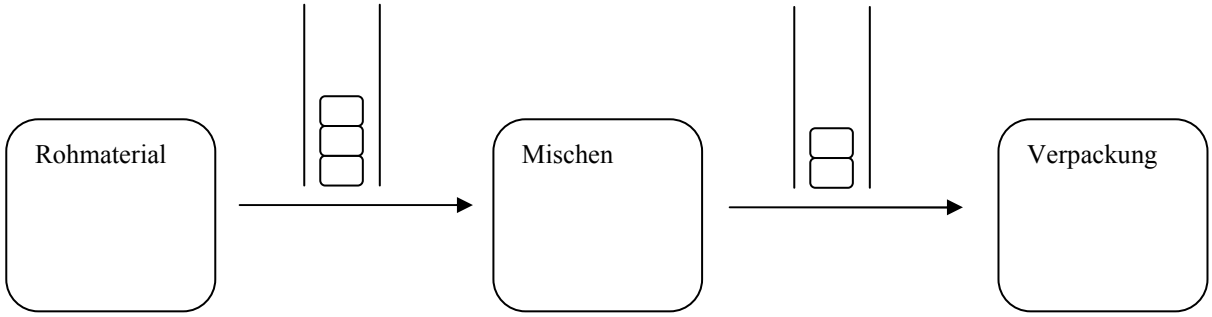
- Die SPS legt bei Einschalten READY auf 1
- WinCC kann die SPS-Aktion mit START=1 auslösen
- START wird mit ACK=1 als Handshakesignal quittiert
- READY geht zeitgleich mit ACK=1 auf Null, und bleibt Null solange die mechanische Aktion an der SP läuft

Die Synchronisierung der losen Kopplung erfolgt über 2 WinCC-Tags :

PUFFER_roh_misch wird mit jedem Rohzubereitungsvorgang um 1 erhöht, mit jedem Mischvorgang um 1 verringert

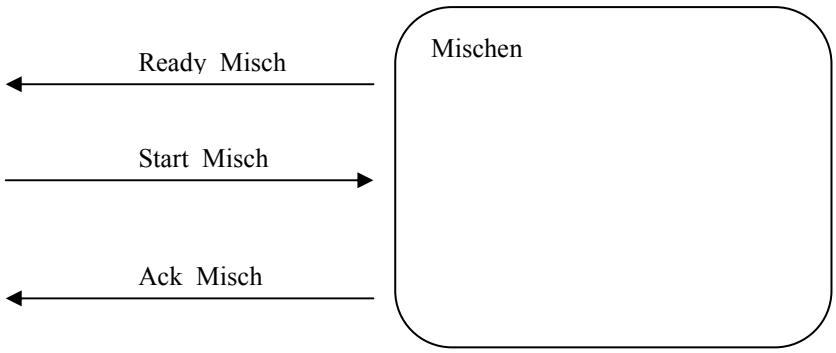
PUFFER_misch_ver wird mit jedem Mischvorgang um 1 erhöht, mit jedem Verpackungsvorgang um 1 verringert

- Wenn ein PUFFER kleiner 1 wird, muß die folgende Station bleiben
- Wenn PUFFER größer 10 wird, muß vorausgehende Station stehen bleiben



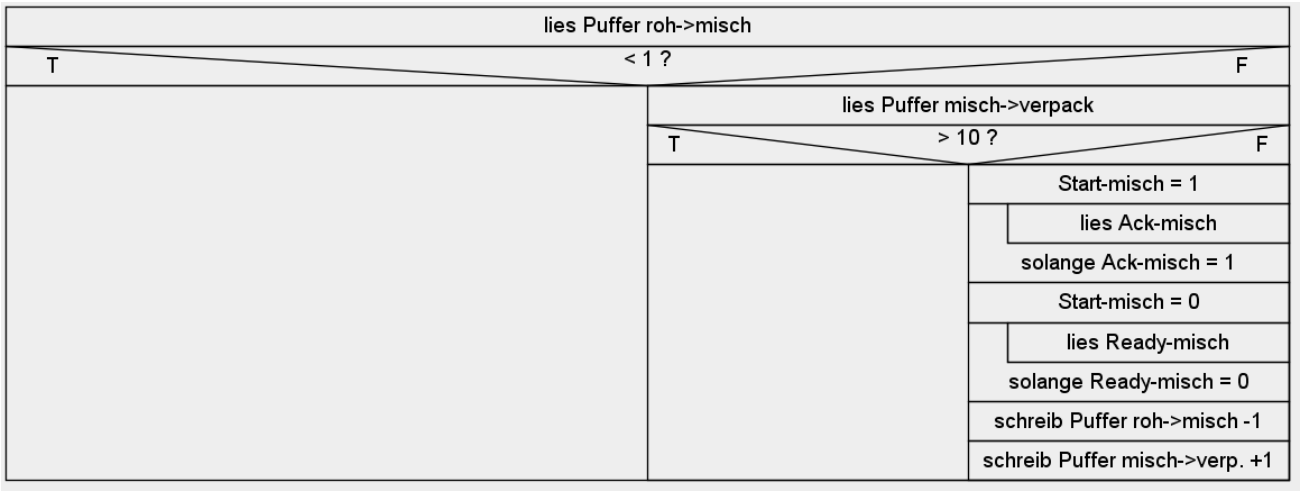
Aufgaben :

1. Zeichnen Sie ein Blockbild der Station Mischen mit allen Ihren Ein-und Ausgängen



2. Entwerfen Sie ein Struktogramm für die einmalige Bedienung der Station „Mischen“ mit dem beschriebenen Handshake, und achten Sie besonders auf die korrekte Realisierung der Puffersteuerung .

   (De Ablauf wird jeweils einmal mit einem Trigger TRIG ausgelöst, Triggerprobleme durch Anlauf oder ähnliches müssen nicht beachtet werden)



3. Codieren Sie das Struktogramm aus 2. in C

```
BYTE c_puffer_r_m, c_puffer_m_v;
BOOL c_start, c_ack, c_ready;
//
c_puffer_r_m = GetTagByte Wait(“PUFFER_roh_misch”);
If (c_puffer_r_m  >= 1 )
        {
        c_puffer_m_v = GetTagByte Wait(“PUFFER_misch_ver”);
        If (c_puffer__v  <10 )
                        {
                        SetTagBitWait(“Start_misch”,1);
                        do
                            {
                            c_ack = GetTagByte Wait(“Ack_misch”);
                            }
                        while ( c_ack = 1);
                        SetTagBitWait(“Start_misch”,0);
                        do
                            {
                            c_ready = GetTagByte Wait(“Ready_misch”);
                            }
                        while ( c_ready = 0);
                        c_puffer_r_m = c_puffer_r_m -1;
                        SetTagBitWait(“PUFFER_roh_misch”,c_puffer_r_m);
                        c_puffer_m_v = c_puffer_m_v+1;
                        SetTagBitWait(“PUFFER_misch_ver”,c_puffer_m_v);
                        }
        }
```