CLEVER system and conveyor control

The binary communication with PLC

(General description for SAS plants using Siemens S7 control)

Updated version from May 2011

PLC system is able to send a stop and release signals to the conveyor controller. On SAS production lines the conveyor means the main chain, which moves all cockpits on the main production line. Theoretically, each station can control its own conveyor, but in your conditions all stations on the main line control one conveyor – the main one (instead of Leipzig, where the situation is a bit different). The conveyor controller is connected to the nearest PLC PC (whichever from the PLC point of view) by a serial line. This connection was implemented by a company SASIT at all SAS plants which are using that (it is necessary to install the serial communication card to the controller and to program it).

From our side – there will be installed the software module "plc_conveyor_controller" – this is the conveyor server, which provides a conveyor control to all PLC stations connected to LAN. (So it is not necessary to connect the conveyor controller physically to each station).

The PLC workstation (when it is configured) can send the stop signal in case of two serious situations:

1. **Sequence error** (Most often it indicates a skip of a cockpit. For example cockpit #1 is finished, an operator forgot to scan the cockpit #2, so when the cockpit #3 is scanned, the sequence error is reported and the conveyor is stopped. Cockpit #2 is not longer moved from the workstation area, so it is possible to process it, a responsible person needs to release a conveyor.)
2. **Not finished cockpit** (some operation is missing on the cockpit (e.g. part check, tightening,…) and an operator is trying to start the next cockpit by a scanning of the next cockpit ID. So the "Cockpit not finished" is reported and the conveyor is stopped,…)
3. **Time expiration** (planned for OC station) – the conveyor will be stopped as well after exceeding of a timer which starts counting beginning from the scan of the previous cockpit (independent whether the previous cockpit was detected with OK or NOK state at the check out station). The timer is programmable because the optimal value will be from plant to plant different.

The conveyor can be released (start again) by different ways, depending on the implementation. Generally are used two following ways: (1) control PLC barcode owned only by a responsible person scanned on the workstation which has sent the stop signal, (2) a key switch placed directly on the conveyor controller.

Each station sends its ID with the stop and release signals. So it is possible to switch on for example a signal lamp on the controller panel to indicate which station(s) caused the line stop.

Communication with the conveyor controller (PLC)

The communication between plc_conveyor_controller (CLEVER software module) and conveyor PLC can be via the serial line or Ethernet tcp/ip. The data content is exactly the same.
The communication uses several-byte telegrams (in both directions) with the following structure:

Byte 0	Byte 1		Byte 2	Byte 3	[Byte 4]	[Byte 5]	[Byte 6]
<ConveyorID>	<AddrLength>	<MID>	<Address_1>	<Address_2>	[<Address_3>]	[<Address_4>]	[…]

Each byte is a binary value (NOT ASCI characters):

<**ConveyorID**>: ID of the conveyor. When there is only one conveyor, the value “1” is used.

<AddrLength+MID>: This Byte consists of two parts – higher bits (AddrLength) and lower bits (MID)
The parameter **AddrLength** defines the length of the workstations in Bytes. If its value is 0 (0000), the address length is 2 Bytes. It is used by older implementations.
<**MID**>: Message ID:

CodeConveyorBlok	0
CodeConveyorUnBlok	1
CodeConnectionCheck	2
CodeConveyorReset	3

Example of that Byte: 0011 0001 – Station address will have 3 Bytes and the command is to unblock (run) the conveyor.

<**Address_1**><**Address_2**>:
Each station sends to a controller its ID. The ID is represented with 1 bit equal to 1 in a mask of address Bytes for example 0000 0000 0000 0001, 0000 0000 0000 0010,... . <Address_1>, <Address_2>, <Address_n> are the first, second, nth 8 bits of the station ID. <Address_x> are the binary values as well (see examples below).

Important for the PLC programming:

If more than 1 station sends the request for a stop of the conveyor, this state is stored in the PLC conveyor controller memory. The conveyor is released when all stations which requested the conveyor stop send the ConveyorUnBlock message (or the message CodeConveyorReset is sent).

The message CodeConveyorReset evokes the release of a conveyor for all stations.

The answer back from the controller side (PLC) must be sent after the each message from PC and has to be in the same format, i.e. the similar bytes:

<ConveyorID><AddrLength+ConveyorState><StationState_1><StationState_2>[..]

ConveyorID and AddrLength are the same as in the received message. ConveyorState is 0 (0000) if the conveyor is stopped or 1 (0001) if the conveyor is running. In address bytes the state of a conveyor is sent in the same rules as the client address. e.g. 0000 0000 0000 0101 means the conveyor is stopped from stations 1 and 3.

The values of sent bytes are binary values (NOT ASCII). E.g. message "1 1 0 2" will be four bytes with a value 0x01, 0x01, 0x00 and 0x02.

The station addresses are implemented as the bit masks, because it is easy to handle and accumulate them using simple bit arithmetic. For example two (or more) stations send the requests to stop the line, so the final state of the register is easy to be calculated:

0000 0000 0000 00**1**0

0000 0000 0**1**00 0000

0000 0000 0**1**00 00**1**0

Following examples are for the default AddrLength 2, which is not specified in the second Byte.

Examples:

Message 1 0 0 1: Request for a stop of conveyor ID 1 from a station ID 0000 0000 0000 0001

Message 1 1 0 2: Request for a release of conveyor ID 1 from a station ID 0000 0000 0000 0010

Example of the full communication:

Message 1 0 0 4: Request for a stop of conveyor ID 1 from a station ID 0000 0000 0000 0100 (station 3)
Answer 1 0 0 4: State 0000 0000 0000 0100 (Conveyor is stopped from a station 3)

Message 1 1 0 4: Request for a release of conveyor ID 1 from a station ID 0000 0000 0000 0100
Answer 1 1 0 0: State 0000 0000 0000 0000 (Conveyor is running)
…

Message 1 0 0 1: Request for a stop of conveyor ID 1 from a station ID 0000 0000 0000 0001 (station 1)
Answer 1 0 0 1: State 0000 0000 0000 0001 (Conveyor is stopped from a station 1)

Message 1 0 0 4: Request for a stop of conveyor ID 1 from a station ID 0000 0000 0000 0100 (station 3)
Answer 1 0 0 5: State 0000 0000 0000 0101 (Conveyor is stopped from stations 1 and 3)

Message 1 1 0 1: Request for a release of conveyor ID 1 from a station ID 0000 0000 0000 0001 (station 1)
Answer 1 0 0 4: State 0000 0000 0000 0100 (Conveyor is stopped from a station 3)

Message 1 0 0 64: Request for a stop of conveyor ID 1 from a station ID 0000 0000 0100 0000 (station 7)
Answer 1 0 0 68: State 0000 0000 0100 0100 (Conveyor is stopped from stations 3 and 7)

Message 1 2 0 0: Request for a check of conveyor state
Answer 1 0 0 68: State 0000 0000 0100 0100 (Conveyor is stopped from stations 3 and 7)

Message 1 3 0 0: Request for a reset of conveyor ID 1
Answer 1 3 0 0: State 0000 0000 0000 0000 (Conveyor is running)

Bit examples:

Byte 0	Byte 1		Byte 2	Byte 3	Byte 4
Conveyor ID	AddrLength	MID	StationAddr_1	StationAddr_2	StationAddr_3
0000 0001	0000	0000	0000 0000	0000 0001	
Conveyor 1	0 = 2 bytes	Stop	Station 1		
0000 0001	0011	0000	0000 0000	0000 0000	0000 0001
Conveyor 1	3 bytes	Stop	Station 1		
0000 1001	0011	0001	0000 0000	0000 0100	0000 0000
Conveyor 9	3 bytes	Run	Station 13		