Potential Areas of Implementation for EasyIO30P http://www.easyio.com/easyio-30p-bn (Draft) rev. 0.0.3

Production Status EasyIO Interactive Meter. System Architecture (Abstract).

The system consists of

1. Sedona device, software components linked in wiresheet to form a client application logic.
2. Client application logic determines: a. Which digital inputs derivatives (ON/OFF time, ON/OFF count) and/or analog inputs involved in a particular case. b. How to handle and upload events to the server. Digital outputs are intend to be controlled remotely from the server, although local override function is available.
   There are 2 possible modes of operations: online and offline.
   In online mode the client application maintains persistent connection with the application on remote server.
   Client has to establish and confirm connection with the server prior switching automatically to online mode.
   Anytime manual override from online to offline mode of operations.
   Manual override from offline to online mode of operations, while confirmed connection exists.
   Automatic fallback to offline mode on connection failure.
   Automatic recovery of failed server connection.
   Automatic recovery of online mode on connection recovered.

   While confirmed connection exists, client communicates to the server regardless of mode of operations.
3.

Case: Parallel Tracking of Expected vs. Actual Production State(s) and Event(s) for OEE input.

Justification:

- Diversified manufacturing environment
- Relaxed internal standards for machine suppliers
- Legacy and/or proprietary machinery
- Machinery without external communication channels
- High cost for machinery integration using native protocols
- High cost and maintenance for third party communication packages
- Distorted and/or conflicting production reports
- Time credit disputes between manufacturing departments

Prerequisites:

- Production schedule application/module* (*just tracking without scheduling option is available)
- Maintenance management application/module
- Ethernet or Wi-Fi connectivity to production environment

Limitations:

- input/output resolution >250 ms

Theory of operations:

Preface:
The system is designed around the simple Expected=Actual principle. If Expected<>Actual then inform, investigate and resolve. In other words principle of resolving conflicted states is at the core of the system. At the same time, system is designed to provide undistorted view for both Expected and Actual sides, thus allowing both sides adjust to reasonable expectations.

Boundaries: Each unit is designated to handle one source of production. Although the source may consists of multiple assembly cells and/or machines, from manufacturing perspective, it has to be defined and handled as an atomic source (can't be broken to smaller sources or sub-sources that are subjects to individual tracking).
To simplify, in this document one machine is equivalent to once source of production.

Typical machine has built-in outputs and/or inputs to control it's operation, in many cases machine may include auxiliary I/O as well. Quite common, for instance, signals in cycle and fault. Objective is to clearly identify these outputs and/circuits, their relation to machine state, level of automation, interdependency and exclusiveness.
Example: the signal INCYCLE is mutually exclusive with the signal FAULT, e.g. if INCYCLE then not FAULT, or if FAULT then not INCYLE, or if not INCYCLE and not FAULT then IDLE. If signal is required but not available through machine automation, signal placed into manual signals category. Manual interaction is required for that category.
State for no signals must be defined as well, for instance IDLE. If power present and communication with the server is not available, unit automatically activate corresponding state. Server application identifies and records internal signal on failure to communicate with the device. Thus system is designed to maintain persistent communication and inform production of expected state or states and indicate any discrepancies that may occur between actual and expected state(s). Production state(s) that can not be automated, must be acknowledged or raised manually.
It is possible not only to receive events but to send events to the server as well. For instance, maintenance and lo supplies events may be raised manually, then received by the server and propagated to the responsible party.

Example of initial requirements gathering and prototyping for abstract machine:
Identified states/signals:

Nr	Signal Name	Stack Light	Source	Input/Data Type	Mode	Priority	Mutually Exclusive with Signal(s)	Schedule
1	AutoInCycle	Green	Machine	Digital	Meter On Timer (sec)	1	3,8,4	Y
2	PartProduced	n/a	Machine	Digital/Pulse	Meter On Counter	n/a	3,8	
3	Fault	Red	Machine	Digital	Meter On Timer (sec)	5	1,2,8	
4	ChangeOver	Blue	Manual PB	Digital	Meter On Timer (sec)	3	6,8,1	Y
5	Maintenance	Yellow	Manual Key+PB	Digital	Meter On Timer (sec)	3	8	Y
6	ShutDown	White	Manual PB	Digital	Meter On Timer (sec)	2	4,8	Y
7	Lo Supplies	Amber	Manual PB	Digital	Meter On Timer (sec)	n/a	n/a	
8	Idle	All Off	Automatic	Memory bit	Meter On Timer (sec)	4	1,2,3,4,5,6	
9	CommFault	All Flash	Automatic	Memory bit	Meter On Timer (sec)	0	n/a	

Fig.1 Signals, Sources, Prioritization table

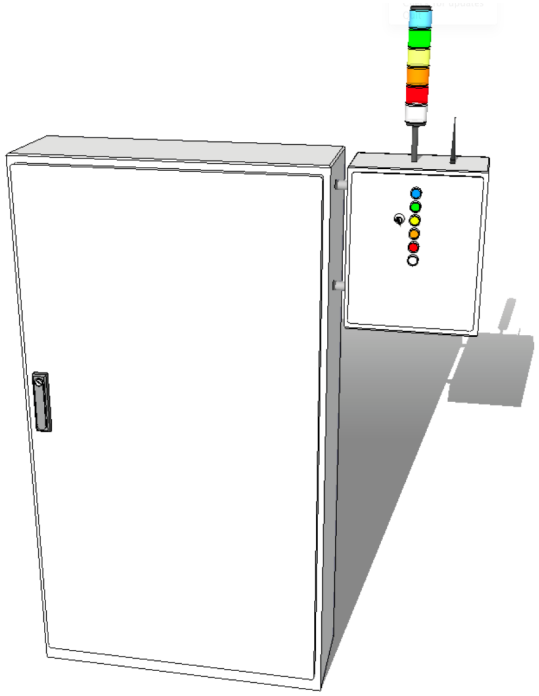

Fig.2 Abstract, electric panel with interactive unit attached on the side.

Examples of operations:

Lets say machine is scheduled for production from 8:00AM until 4:00PM.
Total of 8 hrs, of which
Breaks: 10:00am-10:15am, 2:00pm-2:15pm
Lunch: 12:00pm-12:30pm
Total break time: 1 hr

Change Over is expected after lunch and should last not more than 15 min, e.g. 12:30pm- 12:45pm
Total change over time: 0.25 hr

Thus, having 8-1-0.25= 6.75 hrs expected production availability.
Knowing the production rates (if before and after change over rates are different) or rate for this machine, possible to calculate OEE. Lets say machine has constant production rate of 5 sec/pcs. Therefore a 100% OEE translates as:
(6.75 hrs * 3600 sec)/5 sec/pcs = 4860 pcs produced over scheduled 8 hrs.

Now lets look at our hypothetical case.
At 8:00am the server will activate state (in cycle) expected production and sends it to the client unit.
At that exact moment operator was preparing the machine for the run and no signals were available (idle).
Due to mismatch between the actual state (idle) and the expected state (in cycle) a corresponding light(green) begin to blink. Upon machines transition to (in cycle) state, corresponding(green) light will turn on solid, as a result of match between actual and expected states. As production progress further the partproduced meter begin to increment its value. Unit periodically communicates with the server, sending current values of all meters and receiving updates for output(s). Each communication trigger server to process received data and calculate instant OEE, that can be displayed, for example to andon panel along with part count.

Machine runs for some time then fault occurs. [incycle] and [fault] are 2 mutually exclusive signals, thus having actual state=[fault], expected state=[incycle]. Stack light green[incycle] and red[fault] indicators are blinking.
At certain point operator has cleared the fault and machine transitioned to a no signal state [idle], then having only green[incycle] indicator blinking as a result of actual state=[idle], expected state=[incycle].
Soon machine resumed operations, green indicator turn on solid due to expected=actual=[incycle].

In the next example more complex scenario of parallel states is observed. For instance, after experiencing several concecutive faults operator decided to run machine in degraded performance mode and call for maintenance by pushing the yellow button momentaraly. The [maintenance] signal is received by the server and acknowledged by adding second expected stated=[maintenence]. Thus having: expected state 1 =[incycle], expected state 2=[maintenence] actual state=[incycle] -degraded. Indication: green light is solid on, yellow light is blinking (awaiting maintenance).
When arrived, maintenance resource inserts and turns a personal key to acknowledge(check in).
Now having the following:
expected state 1 =[incycle], expected state 2=[maintenence]
actual state 1=[incycle] -degraded, actual state 2=[maintenence]. Indication: green light is solid on, yellow light is solid on (maintenance in progress). If personal key maintained in on position for more than 1 min, system resets the expected maintenece signal upon key removal (check out) automatically.
The above scenario demonstrates not only tracking capability but also an effective way of handling maintenance calls, while effectively tracking maintenance in progress time.

System designed to effectively fill gaps left by higher priority states with lower priority states, providing contigous overall state. For equal priority states e.g. [changeover] and [maintenance] FIFO (first in first out) queue applied. In any case any parallel states are credited independanly of overall state using this state versus other state(s) method. This effectively resolves conflicts between 2 states when time credited to [incycle] state while [maintenance] is in progress.

Summary of 8 hrs of sheduled production:

Signal	Actual Overall	Scheduled	Actual Specific	Waiting	In Progress	OEE impact	Pcs impact	aOEE impact %	SOEE impact %
[incycle]*	20300 sec	24300 sec (6.75 hrs)	n/a			-4000 sec, total downtime	-800	-16.33	-16.46
[partsproduced]	3400 pcs	4860 pcs @5 sec/pcs	n/a			1460-800=660 (degraded)	-660	-13.47	-13.58
[fault]	600 sec	n/a	n/a			-600 sec	-120	-2.45	-2.47
[changeover]*	700 sec	900 sec	700 sec			+200 sec to available time	vary		
[maintenence]	3000 sec	n/a	4000 sec	1500 sec	2500 sec	-3000 sec	-600	-12.24	-12.34
[shutdown]*	3600 sec	3600 sec	n/a			0	0	0	0
[idle]	400 sec	n/a	400 sec			-400 sec	-80	-1.64	-1.65

Total time	28800 sec	28800 sec (8 hrs)
Available time	24500 sec	24300 sec (6.75 hrs)
OEE	69.39%	69.96%

aOEE –*relative OEE*, calculated using actual total available time over predefined schedule
aOEE = (3400*100) / (24500/5) = 69.39%, where losses distribution is:
-16.33% (total downtime), with distribution:
-2.45% (fault)
-12.24% (maintenance)
-1.64% (idle)
-13.47% (degraded rate).

sOEE –*absolute OEE*, calculated using scheduled available time over predefined shedule
sOEE = (3400*100) / (24300/5) = 69.96%, where losses distribution is:
-16.46% (total downtime), with distribution:
-2.47% (fault)
-12.34% (maintenance)
-1.65% (idle)
-13.58% (degraded rate).

The difference between relative and absolute OEEs indicating how good is the scheduled availability, or in other words how accurate is the fulfilment of scheduled availability.

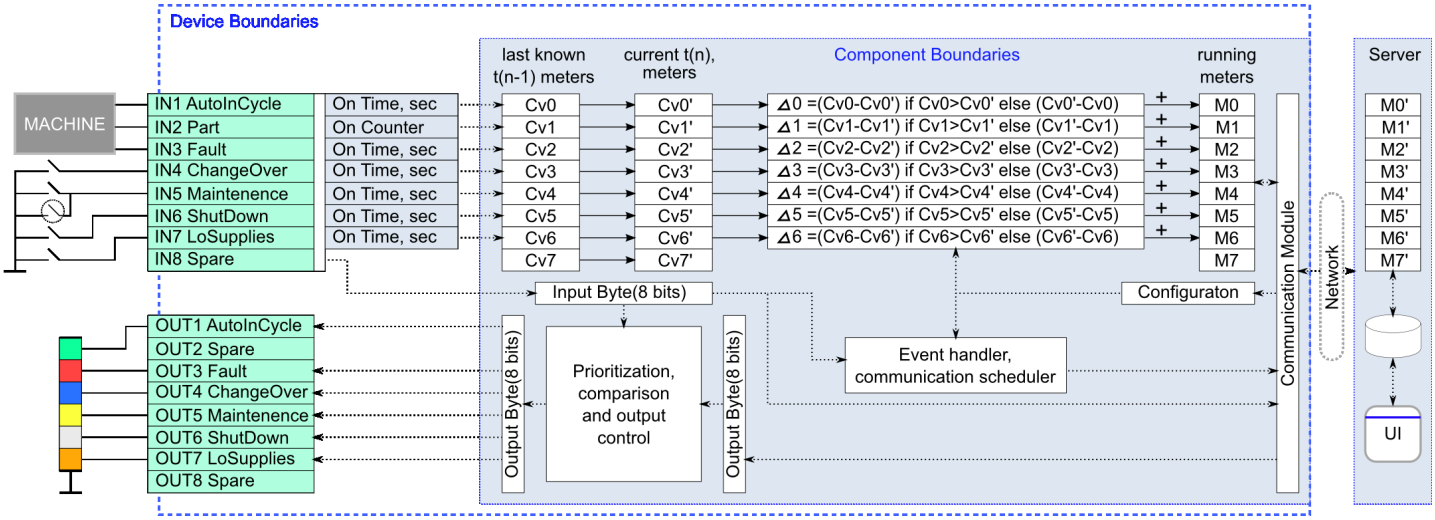

Fig.3 System Communication Diagram

EasyIO30 Interactive Metering data communication protocol over Ethernet TCP/IP.

Device communication mode:	connecting socket (client)
Host application communication mode:	listeneing socket (server)
Packet Fragmentation:	supported
Mtu control:	1500 bytes (adjustible)
Communication mode:	half-duplex
Communication master:	the device (push)

The Device(client) is responcible for establishing a connection with the host application(server) using connecting socket. The Device is responcible for handling connection failure, reconnect using a specific address or, optionally a list of addresses.
The Device is equipped with extended communication recovery mechanism to handle data send and data receive errors and timeouts. In addition, protocol integraty and message length verification included to provide reliable and robust communication.

The Device operates either in connected or disconnected mode. In connected mode the device uses the configuration received from the application during the syncronization phase. According to cofiguration, the device sends data, using schedule or event or both. Every time a data sent to the server, the device is expected a reply within expected time frame and sequence. If no reply received, then error handling mechanism is invoked.

Data messages. There are 2 types of data messages; full and compact.
Full data message format:

```
{stx}<tsn>|<devid>|<cmd>|<config>|<R0>|<R1>|<R2>|<R3>|<R4>|<R5>|<R6>|<R7>|<iobyte>|<Rmsk>|<msglength>{etx}
```

Full data message format utilized for data exchange between device and server. It is used in both directions, from the deveice to the server and from the server to the device.
Although, format is identical for send and receive cases, some element have different intrpritations when send or received.

`{stx},{etx}` - message envelope boundaries
`<tsn>` - transaction sequence number, internal dtata type: sint32
`<devid>` - unique device id, provided by manufacturer, internal datata type: sint32
`<cmd>` - command, 3 character command, string.
Supported commands:

device to server:	SYN - communication syncronization rquest. UPL - data upload. ERR – error.
server to device:	SET - set device' cofiguration and/or registers, and/or outputs, reply to SYN. NXT – reply to run next device cycle, reply to UPL, uses compact format.

`<config>` - configuration data cosists of 2 elemens :< <mode><poll time ms> >
`<mode>` - 1 character, "E" -event or "P" - periodic poll
`<poll time ms>` - range:250 – 360000000 (ms), internal data type: sint32
`<R0>`
…
`<R7>` - communication registers, internal data type: sint32.
In the device to server message, a register represents either current running meter value or value from analog input. Interpreted by a corresponding bit in <Rmsk>, e.g. R0 – bit0, values: 0-meter, 1-analog.
In the server to device message, a register represents either a new value to set for a corresponding meter or analog value to propagate to analog output. Interpreted by a corresponding bit in <Rmsk>, e.g. R0 – bit0, values: 0-meter, 1-analog (max 4 analog + 2 OC outputs).

<iobyte> - input or output byte, internal data type sint32 to binary, lo 8 bits. When sent by the device, represents current state of 8 digital inputs. When received by the device, represents a new states for 8 digital outputs, except when empty.

<rmsk> - type of registers mask, internal data type sint32 to binary, lo 8 bits. Controls how registers are handled. Each bit corresponds to a register, e.g. bit0 – R0, bit1 – R1 etc. Value 0(false) sets meter mode of operations for that register. Value 1(true) sets analog mode of operation for that register. Only 6 analog outputs are available.

<msglength> - total number of bytes in envelore, excluding {stx} and {etx} bytes, internal data type sint32.

Protocol summary. Lightweight EasyIO30 Interactive Metering data communication protocol developed to:

1. Provide data acqusition method for 8 inputs, either analog or digital meters or any combination of both types.
2. Provide control of 8 digital outputs and 6 analog outputs.

Examples of full data message communication:

```
device sent:          {stx}0|98765432|SYN|P10000|0|0|0|0|0|0|0|0|0|44{etx}
device received:      {stx}0|98765432|SET|E10000|0|0|0|0|0|0|0|0|0|44{etx}

device sent:          {stx}200|98765432|UPL|E10000|26|1|0|9999|0|0|0|14|2|0|51{etx}
device received:      {stx}200|98765432|SET|E10000||||0||||||0|38{etx}
```

Examples of compact data message communication:

```
device sent:          {stx}250|98765432|UPL|E10000|31|15|0|0|0|0|0|34|2|0|49{etx}
device received:      {stx}250|98765432|NXT|19{etx}
```

Example of handling invalid received message(s): *- Not Implemented!*

```
device received:      {stx}487|98765432|SET|10000|0|0|0|0|0|0|0|0|0|0|45{etx} – invalid configuration!
Device sent:          {stx}488|98765432|ERR|E10000|31|15|0|0|0|0|0|34|2|0|49{etx}
```

Note! For fach invalid/unhandled message received, the ERR message from device is sent immediately. The ERR message from device has to be handled as UPL type of message. Example:

```
device sent:          {stx}487|98765432|UPL|E10000|31|15|0|0|0|0|0|34|2|0|49{etx}
device received:      {stx}487|98765432|SET|10000|0|0|0|0|0|0|0|0|0|0|45{etx} – invalid configuration!
device sent:          {stx}488|98765432|ERR|E10000|31|15|0|0|0|0|0|34|2|0|49{etx}
```

Home Automation with EasyIO. System Architecture (Abstract). To Do.....

Home Automation

Utilities

Electric — *HVAC permissive controll and scheduling.*
Lighting control, automatic and/or sheduled.
Water heating permissive contol and scheduling.
Emergency and on demand shutoffs.
Usage analysis, historical trends/peaks, comparisons. Audits.

Gas

Water — *Sheduled automated lawn irrigation with wather report/forcast feedback*
Emergency(flood/rapture) and on demand shutoffs.
Usage analysis, historical trends/peaks, comparisons. Audits.

Events

Security
Fire and Flood

Case: Small store/franchise operations. System Architecture.

Purpose of the system. The primary objective of the system is to reduce energy waste. The waste is a result of energy being consumed by the equipment while not in production. The system provides felexible and open remote interface for handling peaks in production demand, thus alowing proactively, or on sheduled bases, or manually, to control transition of equipment from energy save standby mode to production ready mode remotely. Due to some inertia assosiated with bringing a particular type of equipment from standby to operation mode, effective control mechanism possible trough production forecast or schedule or both. At any time the system can be overriden or bypassed by the local resource, responcible for operating the equipment. The basic temperature quality sampling and time in operation ready and/or bypass modes tracking is provided. The latter, in correlation with the (POS) order processing data, enables an excellent fine-tuning input of schedules or events for seamless remote handling of expected equpment availability.

System's physical design, infrastructure and hardware requirements:

1. Ethernet TCP/IP ready Sedona framework device (EasyIO30).
2. Equipment wiring plugin injector with terminals.
3. Control panel, enclosure with switches, indicators and sound horn, 24 VDC or AC power supply.
4. Remote Server application (host system) operational.
5. Local Internet access.

System's capacity: Each device is capable of handling up to 3 units of production eqipment.
The following signals are applicable, per unit of equpmnet:

1. equpmnet is on/off
2. equpmnet is in manual mode (system bypass)
3. equpmnet's current operation temperature

Theory of operations:

Device connected with each equipment using digital and anolog I/Os through plugin injector. On disconnect, the injector alowes equipment operate in manual mode (original mode of operation) only. With device being connected to the equipment there are 2 modes of operations available; a). Manual (System Bypass) or b). Automatic (Econo) modes. Each of 2 modes have 2 communication state submodes; online and offline. Therefore total of 4 variations are possible:

1. Bypass-Offline
2. Bypass-Online - quality and state(s) time tracking only
3. Automatic-Offline
4. Automatic-Online - enable equipment remotely + quality and state(s) time tracking

Offline operations. The offline submode is an indication of no connectivity with the remote server. On online to offline transition and while offline, the local system notifyes operator every 10 minutes by 1 short beep sound. During the offline submode no data communicated with the server. The server's application is tracking the offline state time for its entire duration. In offline occurred while in Automatic, All device's digital control outputs to the equipment will maintain their states while device is offline. If any of the device's output(s) prevent equpment from being available, operator must switch the corresponding equipment to the manual mode (system bypass).

Bypass operations, performed only when system is in contradiction with the actual demand for production. Regardless of communication state, operator is reminded every 10 minutes about bypass being present, using 2 short beep sounds and indication light. Ultimate goal is to minimaze or not to have bypass cases it all.

Multiple audioble notifications delivered in sequence , for instance 1 short + 2 short beep sounds are translated as "system offline, bypass present".
Online operations. While online, the system is actively tracking current state times and quality data. Moreover, when Automatic mode is enabled, the system activates access to equipment remote control. The control algorithm itself is not

in the scope of the system. The purpose of the system is to provide a common control interface for external system(s) or application(s), and collect equipment's historical data. Collected data is intended for use as a statistical intelligence input channel to external system or application, hosting the controling algorithm(s). It is possible for the remote controlling system to enable or disable the equipment. Every time the controlling system enables the euipment for production, operator(s) are notified 30 sec. in advance for safety to clear with 1 long beep. Only in Online Operations mode the remote control is possible.

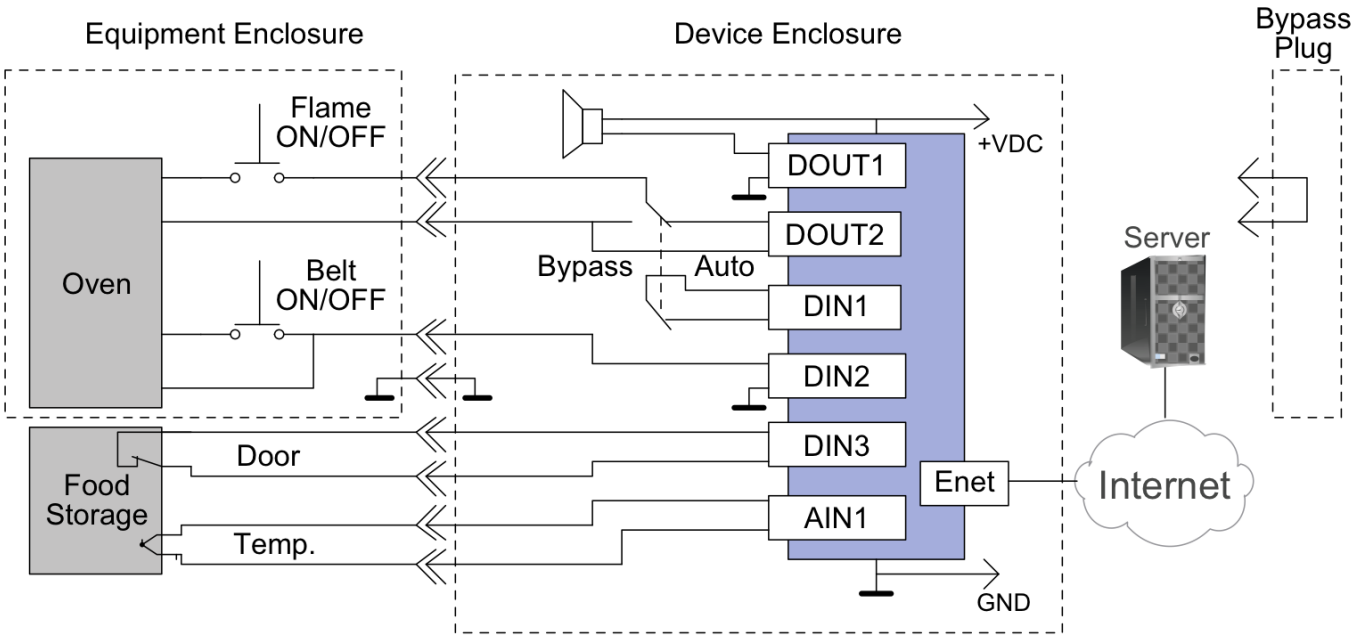

Fig.4 Online Communication and Bypass methods.

Hardwired Bypass. When reliability of the Bypass switch is questionable, the Bypass plug loopback provides an ultimate solution for restoring the original mode of operations.

Example of operations:

For instance there is a hypothetical store with 2 pizza ovens. It is located in a typical suburban area with some small to medium businesses near by. Average maximum throughput of one oven is 1 pizza every 3 minutes, or 20 pizzas/1 hour. Each oven consumes x BTUs of natural gas per hour at nominal operating temperature. Demand is expected to revolve around typical meal patterns, e.g. 5PM-7PM for dinner and 20PM-21PM for supper. Lets apply a hypothetical average demand for product distributed over 4PM – 11PM (work days hours of operations) interval. The data from POS system servs as an input Fig.5. Plotting graph Fig.6, using ½ hour resolution for order volume.

Nr	Time Interval	Orders Received	Nr	Time Interval	Orders Received
1	16:00-16:30	8	8	19:30-20:00	14
2	16:30-17:00	9	9	20:00-20:30	13
3	17:00-17:30	17	10	20:30-21:00	11
4	17:30-18:00	18	11	21:00-21:30	7
5	18:00-18:30	15	12	21:30-22:00	5
6	18:30-19:00	10	13	22:00-22:30	6
7	19:00-19:30	9	14	22:30-23:00	4

Fig.5

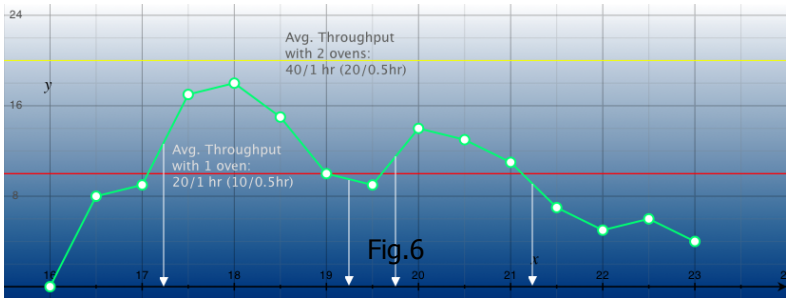

Fig.6

Now it is possible to compose a expected demand schedule, thus having:

1. Enable oven1 at 16:00
2. Enable oven2 at 17:15, here some delay introduced to compencate for inertia, not all 18 orders received at 17:30, instead it is a distribution of 18 over 30 minutes interval (17:00-17:30). In other words, there is always an offset from the time order received, until order enters oven
3. Disable oven2 at 19:15, alows an additional safety gate for finishing volume of orders between 18:30 and 19:00, although 19:00 would probably be more economical.
4. Enable oven2 at 19:45 to compencate for expected increase in demand.
5. Disble oven2 at 21:15, althought disabling at 21:00 gives some extra energy savings.

The shedule may be implemented using discrete or analog models. Former is simply configured using ON/OFF over period of time, while latter offers greater flexibility by interpreting expected input demand over time into multiple ON/OFF signals using multiple configurable threshholds.

Get actual settings and wiring diagram for 2 oven site operations:

Nr	Signal	Type	Name	Accumulator/ koefficient	Register
1	DI1	Digital Input	**Oven1 Burner is ON**=1/OFF=0	Time ON, sec	RO
2	DI2	Digital Input	**Oven1 Bypass is ON**=1/OFF=0	Time ON, sec	R1
3	DO1	Digital Output	**Enable Oven1** ON=1/OFF=0	n/a	n/a
4	DI3	Digital Input	**Oven2 Burner** is ON=1/OFF=0	Time ON, sec	R2
5	DI4	Digital Input	**Oven2 Bypass** is ON=1/OFF=0	Time ON, sec	R3
6	DO2	Digital Output	**Enable Oven2** ON=1/OFF=0	n/a	n/a
5	DI5	Digital Input	**Food Storage is Open**=1/Closed=0	Time ON, sec	R6
6	AI1	Analog Input	**Food Storage Temperature**	K=0.01, F°	R7
7	DO8	Digital Output	Sound Horn	n/a	n/a
8	Registers Mask =00100000=0x186A0				
9	Configuration =E10000 (Event/10sec. Pushed Poll)				

Technical Notes:

1. Getting <rmask> - byte for server to device reply packet. The <rmask> value defines a type of operations for 8 communication register in device software component.
   Each corresponding bit [0-7] sets operation either to meter type(default)=false or analog type=true modes.
2. Retreival of configured <rmask> done through the following query:

```
DeviceID             FieldNr ChNr   ChDirection      ChType           IsActive
-------------------- ------- ------ ---------------- ---------------- --------
91119083             8       2      Input            Analog           1
91119083             6       1      Input            Analog           1

(2 row(s) affected)
```

```
select IsActive, SUM(POWER(2,FieldNr-1)) as rmask from dbo.tblDeviceFields where
DeviceID       =91119083
and FieldNr    is not null
and ChNr       is not null
and ChDirection='Input'
and ChType     ='Analog'
and IsActive   =1
group by IsActive

IsActive rmask
-------- -----------
1        160

(1 row(s) affected)
```

   On configuration change, retreived <rmask> value is forwarded to the tblQueue table column Mask, to be downloaded to a device.
3. Digital state of input(s) can be transferred using communication register(s). For instance, boolean to integer conversion linked to non-metered register schenario.
4. Calculation of expected time for each bit in `OutputStateMux` vlaue sent in reply message.

```
select top 20
OutputStateMux,
case when  (OutputStateMux & CAST(POWER(2, 7) as integer))=CAST(POWER(2, 7) as integer) then '1'
else '0' end as bit7,
case when  (OutputStateMux & CAST(POWER(2, 6) as integer))=CAST(POWER(2, 6) as integer) then '1'
else '0' end as bit6,
case when  (OutputStateMux & CAST(POWER(2, 5) as integer))=CAST(POWER(2, 5) as integer) then '1'
else '0' end as bit5,
case when  (OutputStateMux & CAST(POWER(2, 4) as integer))=CAST(POWER(2, 4) as integer) then '1'
else '0' end as bit4,
case when  (OutputStateMux & CAST(POWER(2, 3) as integer))=CAST(POWER(2, 3) as integer) then '1'
else '0' end as bit3,
case when  (OutputStateMux & CAST(POWER(2, 2) as integer))=CAST(POWER(2, 2) as integer) then '1'
else '0' end as bit2,
case when  (OutputStateMux & CAST(POWER(2, 1) as integer))=CAST(POWER(2, 1) as integer) then '1'
else '0' end as bit1,
case when  (OutputStateMux & CAST(POWER(2, 0) as integer))=CAST(POWER(2, 0) as integer) then '1'
else '0' end as bit0
from dbo.tblRaw
group by OutputStateMux

OutputStateMux bit7 bit6 bit5 bit4 bit3 bit2 bit1 bit0
-------------- ---- ---- ---- ---- ---- ---- ---- ----
NULL           0    0    0    0    0    0    0    0
0              0    0    0    0    0    0    0    0
1              0    0    0    0    0    0    0    1
3              0    0    0    0    0    0    1    1
128            1    0    0    0    0    0    0    0
129            1    0    0    0    0    0    0    1
131            1    0    0    0    0    0    1    1

(7 row(s) affected)
```

Objective: to have time while the bit(n) is on, accumulated or summed over certain interval.

5. Add comparison analysis for similar devices, controllers. Emphasise on infrastructure that required to support demonstated case versus competetive devices, controllers.

Appendix **A** Case Small store/franchise operations. Communication Sequence Diagram