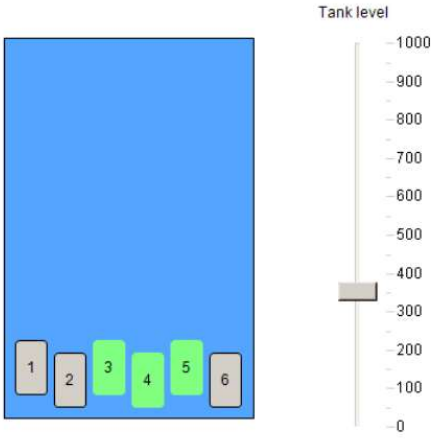
Six Pumps Control



Control of six pumps inside tank. Number of pumps in operation depends on the liquid level.

When tank is full, all pumps must run in order to operate with the highest possible pump capacity.

The pumps run in alternating operating mode i.e.:

- the pump with the lowest accumulated operating time must start when higher liquid level requires additional pump.
- the pump with the longest accumulated operating time stops when liquid level become lower
- after defined period of time '*AlternationTimeSec*' in *AlternatePump* functio*n,* theh pump with the longest accumulated operating time stops and the pump with lowest accumulated operating time starts

Alternating operation ensures that pumps are used for the same amount of time and this reduce maintenance and service costs.

```
TYPE PumpState:
       (STOP, RUN, ALARM, SERVICE):= STOP;
END_TYPE


TYPE PumpType:
STRUCT
       RunState                        :      PumpState := PumpState.STOP;
       RunTotalMinutes                 :      DWORD:=0;
       RunLastStartSec                 :      WORD:=0;
       RunSeconds                      :      WORD:=0;      // count up to 60
       RunTimerSeconds                 :      TON;          // count up RunSeconds
END_STRUCT
END_TYPE


TYPE PumpStation :
STRUCT
       PumpsArray                      :      ARRAY[1..GVL.MAX_PUMPS] OF PumpType;
       NoPumpsRun                      :      INT;   // number of running pumps
       Level                           :      WORD;  // water level in the tank
END_STRUCT
END_TYPE
```

```
PROGRAM MAIN
VAR
	PumpTank	:	PumpStation;		// one pump station ceontains all pumps
END_VAR

// update counters
UpdateCounters(Pumps:=PumpTank.PumpsArray);

// pump alternating time is 3600 seconds = 1 hour.
// can be changed to a lower time  when testing
AlternatePump(AlternationTimeSec:=3600, PumpTank:=PumpTank);

// update nomber of running pumps
PumpTank.NoPumpsRun := NoOfRunningPumps(Pumps:=PumpTank.PumpsArray);

// find the required pumps to run depending on the level in the tank
LevelControl(751, 1000, 6, PumpTank);
LevelControl(601, 750, 5, PumpTank);
LevelControl(451, 600, 4, PumpTank);
LevelControl(301, 450, 3, PumpTank);
LevelControl(151, 300, 2, PumpTank);
LevelControl(100, 150, 1, PumpTank);
LevelControl(0, 99, 0, PumpTank);
```

```
FUNCTION NoOfRunningPumps : INT
VAR_IN_OUT
	Pumps		:	ARRAY[1..GVL.MAX_PUMPS] OF PumpType;
END_VAR
VAR
	Idx		:	INT;
	No		:	INT:=0;		// counter of running pumps
END_VAR

(*
Count number of running pumps
*)

FOR Idx:=1 TO GVL.MAX_PUMPS DO
	IF Pumps[Idx].RunState = PumpState.RUN THEN
		No:=No+1;
	END_IF
END_FOR

NoOfRunningPumps := No;
```

```
FUNCTION AlternatePump : BOOL
VAR_INPUT
	AlternationTimeSec  :	DWORD;		// time a pump has to run before alternating
END_VAR
VAR_IN_OUT
	PumpTank		:	PumpStation; // address pointer to the STRUCT outside the
function
END_VAR
VAR
	Idx			:	INT;		// loop index
END_VAR

(*
FC for alternating between pumps.
AlternationTimeSec -  Time until pump alternation
*)

// only alternate between pumps if at least one, but not all pumps are running
IF PumpTank.NoPumpsRun > 0 THEN
	IF PumpTank.NoPumpsRun < GVL.MAX_PUMPS THEN
		// check all pumps
		FOR Idx:=1 TO GVL.MAX_PUMPS DO
			IF AlternationTimeSec<PumpTank.PumpsArray[Idx].RunLastStartSec AND
			   PumpTank.PumpsArray[Idx].RunState=PumpState.RUN THEN

			  // first stop the pump that is running for the longest time
			  StopPump(Pump:=PumpTank.PumpsArray[Idx]);

			  // then start another pump
			  FindAndStartPump(Pumps:=PumpTank.PumpsArray);

			EXIT;  // only alternate to one pump. Another pump will be checked in next
			       // scan
			END_IF

		END_FOR
	END_IF
END_IF
```

```
FUNCTION FindAndStartPump : BOOL
VAR_IN_OUT
	Pumps		:	ARRAY[1..GVL.MAX_PUMPS] OF PumpType;
END_VAR
VAR
	Idx			:	INT;
	RunTime		:	DWORD;
	PumpIndex	:	INT;
END_VAR

RunTime := 9999999;		// set initial high tiem to find a low run time
PumpIndex := 1;

// find a pump in stop state with the lowest run time
FOR Idx:=1 TO GVL.MAX_PUMPS DO
	IF RunTime > Pumps[Idx].RunTotalMinutes AND Pumps[Idx].RunState=PumpState.STOP THEN
		RunTime:=Pumps[Idx].RunTotalMinutes;
		PumpIndex:=Idx;
	END_IF
END_FOR

// start pump
StartPump(Pump:=Pumps[PumpIndex]);
```

```
FUNCTION FindAndStopPump : BOOL
VAR_IN_OUT
	Pumps		:	ARRAY[1..GVL.MAX_PUMPS] OF PumpType;
END_VAR
VAR
	Idx			:	INT;
	RunTime		:	DWORD;
	PumpIndex	:	INT;
END_VAR

// set an initial lwo time to find the highest run time
RunTime := 0;
PumpIndex := 0;


// find the pump in run state with the longest accumulated run time
FOR Idx := 1 TO GVL.MAX_PUMPS DO
	IF RunTime < Pumps[Idx].RunTotalMinutes AND Pumps[Idx].RunState = PumpState.RUN THEN
		RunTime := Pumps[Idx].RunTotalMinutes;
		PumpIndex := Idx;
	END_IF
END_FOR

// stop the pump
IF PumpIndex > 0 AND PumpIndex <= GVL.MAX_PUMPS THEN
	StopPump(Pump:=Pumps[PumpIndex]);
END_IF
```

```
FUNCTION LevelControl : BOOL
VAR_INPUT
	LevelLow		:	WORD;		// liquid level in tank must be above this level
	LevelHigh		:	WORD;		// liquid level in tank must be below this level
	NoPumpsReq		:	INT;		// number of pumps required to run in the liquid
							// level range
END_VAR
VAR_IN_OUT
	PumpTank		:	PumpStation;	// address pointer to the STRUCT outside the
							// function
END_VAR
VAR
	PumpControl		:	INT:=0;
END_VAR

(*
This FC starts or stops a pump depending on the level in the tank
*)

// check level. Does a pump need to be started due to a high liquid level?
IF PumpTank.Level>=LevelLow AND PumpTank.Level<=LevelHigh THEN
	IF NoPumpsReq>PumpTank.NoPumpsRun THEN
		// negative if one more pump needed
		PumpControl:=NoPumpsReq-PumpTank.NoPumpsRun;
	END_IF
END_IF

// check level. Does a pump need to be stopped due to a low liquid level?
IF PumpTank.Level>=LevelLow AND PumpTank.Level<=LevelHigh THEN
	IF NoPumpsReq<PumpTank.NoPumpsRun THEN
		// positivve if a pump has to be stopped
		PumpControl := NoPumpsReq-PumpTank.NoPumpsRun;
	END_IF
END_IF

// stop a pump if needed
IF PumpControl<0 THEN
	FindAndStopPump(Pumps:=PumpTank.PumpsArray);
END_IF

// start a pump if needed
IF PumpControl>0 THEN
	FindAndStartPump(Pumps:=PumpTank.PumpsArray);
END_IF
```

```
FUNCTION StartPump : BOOL
VAR_IN_OUT
	Pump	:	PumpType;
END_VAR
VAR
END_VAR

// start pump
Pump.RunState := Pumpstate.RUN;


FUNCTION StopPump : BOOL
VAR_IN_OUT
	Pump		:	PumpType;
END_VAR
VAR
END_VAR

// pump will stop after 30 sec delay to avoid unnecessary start/stop

IF Pump.RunLastStartSec > 30 THEN
	Pump.RunState := PumpState.STOP;
	Pump.RunLastStartSec := 0;
END_IF


FUNCTION UpdateCounters : BOOL
VAR_IN_OUT
	Pumps		:	ARRAY[1..GVL.MAX_PUMPS] of PumpType;
END_VAR
VAR
	Idx		:	INT;		// loop counter
END_VAR

FOR Idx := 1 TO GVL.MAX_PUMPS DO
	Pumps[Idx].RunTimerSeconds(IN:=NOT Pumps[Idx].RunTimerSeconds.Q, PT:=T#1S);

	// only update when pump in RUN state
	IF Pumps[Idx].RunState = PumpState.RUN THEN

		// check if second timer ended
		IF Pumps[Idx].RunTimerSeconds.Q THEN
			Pumps[Idx].RunSeconds := Pumps[Idx].RunSeconds + 1;
			Pumps[Idx].RunLastStartSec := Pumps[Idx].RunLastStartSec + 1;

			// update minutes
			IF Pumps[Idx].RunSeconds >= 60 THEN
				Pumps[Idx].RunTotalMinutes := Pumps[Idx].RunTotalMinutes + 1;
				Pumps[Idx].RunSeconds:=0;
			END_IF

		END_IF
	END_IF
END_FOR
```