ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra řízení

Řídicí systém vybraných částí cementárny

květen 2010

Autor: **Bc. Peter Nanky**

Vedoucí práce: **Doc.Ing. Jan Bílek, CSc.**

Poděkování

Děkuji mému konzultantovi Doc.Ing. Janu Bílkovi, CSc., za jeho cenné rady, čas a odbornou pomoc, kterou mi poskytoval při tvorbě této práce, tak i kolegům Ing. Janu Strašíkovi, Ing. Jiřímu Novotnému, Ing. Janu Mlíčkovi a Ing. Romanu Sládkovi za pomoc při řešení problémů při samotné práci na projektech.

Čestné prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze, dne ……. …….

podpis

Anotace:

Tento projekt sa zaoberá návrhom a tvorbou riadiaceho systému cementární. Hlavným prínosom praktickej časti tohto projektu je vytvorenie funkčného riadiaceho systému pre zvolené technologické časti dvoch cementární, založeného na štandardoch Siemens PCS 7 a CEMAT. Táto práca vytvára základný prehľad obecného technologického procesu výroby cementu a popisuje funkciu riadiaceho systému pre zvolené technologické časti. Primárne sa práca zaoberá špecializovanými neštandardnými funkciami a riešeniami použitými počas vývoja tohto riadiaceho systému.

Summary:

This final project deals with design and creation of control systems for cement factories. Main contribution of practical part of this project is creating functional control systems for selected technological parts of two cement factories, based on Siemens PCS 7 and CEMAT standards. This text creates basic overview of common cement producing technology process and describes functions of control system for selected technology parts. It primarily describes specialized non-standard functions and solutions used in the process of developing this control system.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Peter Nanky**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Řídicí systém vybraných částí cementárny**

Pokyny pro vypracování:

1. Vytvořte řídicí logiku pro zadané části technologie cementárny Astana.
2. Vytvořte vizualizaci k technologii z prvního bodu.
3. Navrhněte optimalizované řízení automatizované laboratoře v cementárně Radotín pro rychlejší zpracování více vzorků.
4. Vytvořte detailnější vizualizaci laboratoře.
5. Otestujte výsledek bodů 3 a 4 na reálné technologii v praxi.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Doc.Ing. Jan Bílek, CSc.

Platnost zadání: do konce letního semestru 2010/2011

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 8. 2. 2010

Obsah

1. Úvod ... 6

1.1 Siemens SIMATIC PCS 7 ... 6

1.1.1 Automatizačný systém AS ... 7

1.1.2 Operátorský systém OS ... 8

1.2 CEMAT ... 9

1.3 Postup výroby cementu ... 10

2. Projekt Astana - Expedícia ... 13

2.1 Rozbor problému ... 13

2.2 Realizácia ... 16

2.2.1 Sekvencia plnenia zásobníku ... 16

2.2.2 Trojpolohová klapka ... 19

2.2.3 Zobrazenie a zadávanie času ... 20

2.2.4 Vizualizácia ... 24

3. Projekt Astana - Drvenie a transport suroviny ... 25

3.1 Rozbor problému ... 26

3.2 Realizácia ... 27

3.2.1 Komunikačný blok U_EXT_DEVICE ... 27

3.2.2 Rozdelenie technológie na skupiny ... 32

3.2.3 Skupina 10G10 - Vápenec + hlina ... 32

3.2.4 Skupina 10G20 - Transport suroviny ... 33

3.2.5 Skupina 10G30 - Transport čistého vápenca do sila ... 34

4. Projekt Radotín - Automatické laboratórium ... 35

4.1 Pôvodný stav ... 35

4.2 Rozbor problémov ... 37

4.3 Realizácia ... 39

4.3.1 Nová koncepcia systému ... 39

4.3.2 Riadenie laboratória ... 42

4.3.3 Prijímacia stanica ... 42

4.3.4 Odsávanie RIBO ... 44

4.3.5 Riadenie presunu kalíškov ... 45

4.3.6 Súčasný beh hlavných sekvencií a zdieľanie prostriedkov ... 49

4.3.7 Karusel ... 50

4.3.8 Posun času ... 58

4.3.9 Sekvencia na nastavenie parametrov pre mlyn a lis ... 60

4.3.10 Komunikácia so zariadeniami ... 61

4.3.11 Vizualizácia laboratória ... 64

5. Oživenie a testovanie laboratória ... 68

6. Záver ... 70

Zoznam použitej literatúry ... 71

Zoznam použitých skratiek ... 72

Prílohy ... 73

1. Úvod

Táto práca sa zaoberá popisom tvorby častí riadiacich systémov dvoch cementární, na ktorých som sa podieľal v rámci práce vo firme SIDAT. Prvá kapitola uvádza základné informácie o použitých systémoch Siemens PCS 7 a CEMAT, ako aj základný pohľad na priebeh technologického procesu výroby cementu. Druhá a tretia kapitola obsahuje popis funkcie a tvorby riadiaceho systému pre expedíciu cementu resp. drvenie a transport suroviny pre cementáreň v Astane. Štvrtá kapitola popisuje funkciu, tvorbu a realizáciu nového riadiaceho systému automatizovaného laboratória v cementárni Radotín, ktorého nasadenie a oživenie je popísané v poslednej piatej kapitole.

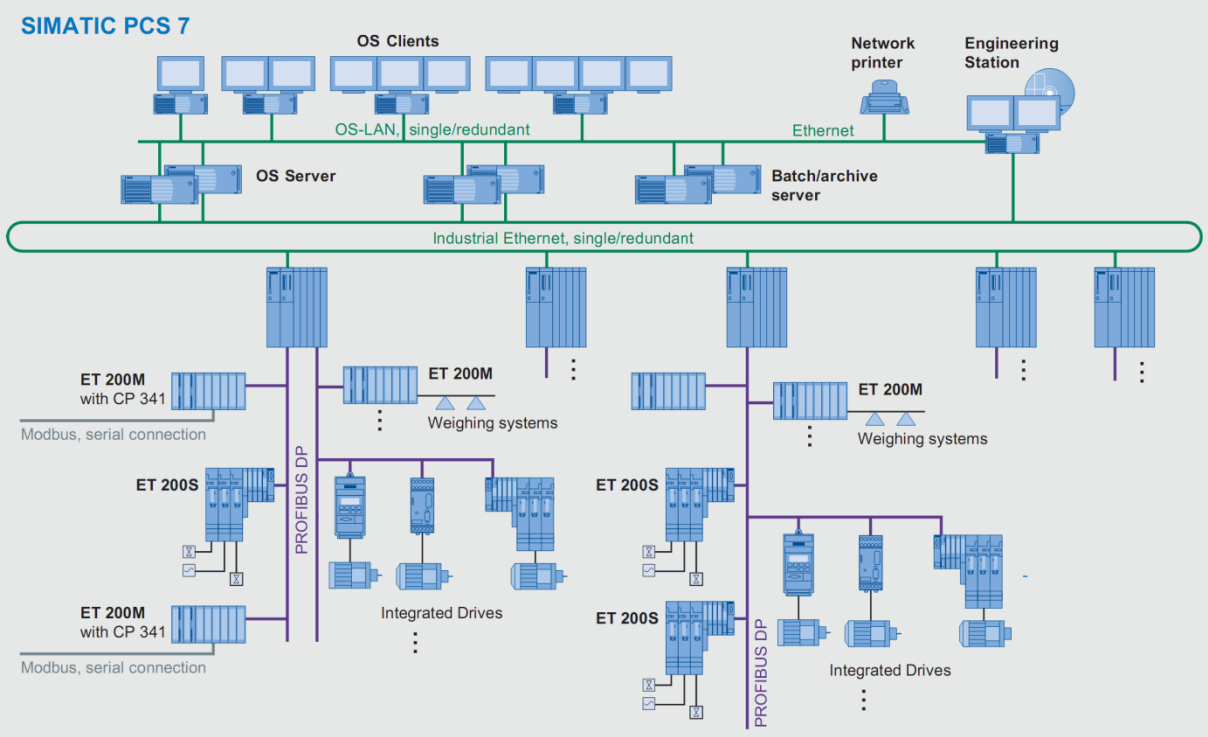
1.1 Siemens SIMATIC PCS 7

SIMATIC PCS 7 je procesný riadiaci systém, založený na štandardných hardvérových a softvérových komponentoch, s modernou architektúrou typu Client-Server, s prepojením na systém riadenia podniku. Obsahuje inžiniersky, operátorský, automatizačný, údržbový systém, komunikáciu a distribuované periférie.

Inžiniersky systém je tvorený správcom SIMATIC Manager. Pri programovaní je možné využiť všetky funkcie, ktoré sú dostupné v prostredí Step 7 pre diskrétne riadenie. Navyše sú k dispozícii jazyk SCL (Structured Control Language), zobrazenie procesných objektov POV (Process object view) a programovanie pomocou kontinuálnych funkčných diagramov CFC (Continuous function chart). Pre dávkovú výrobu je možné využiť programovanie pomocou sekvenčných funkčných diagramov SFC (Sequential function chart).

Automatizačné systémy procesného riadiaceho systému SIMATIC PCS 7 sú založené na vybraných komponentoch z rodiny SIMATIC S7-400. Ich výhody spočívajú v modulárnom návrhu, veľkej možnosti rozširovania, robustnosti, možnosti jednoduchého alebo redundantného zapojenia a ľahkého spojenia s centrálnymi alebo distribuovanými I/O. Pre Systém PCS 7 existuje množstvo periférií od distribuovaných I/O modulov ET200M, ET 200S, ET 200iSP s širokou paletou signálových a funkčných modulov po inteligentné distribuované poľné/procesné systémy a operátorské terminály. Ďalej môžu byť do SIMATIC PCS 7 integrované cez PROFIBUS s využitím add-on blokov I/O moduly ako systémy pre správu motorov SIMOCODE, frekvenčné meniče MICROMASTER 3/4 alebo vážiace systémy SIWAREX M/U/FTA/FTC a iné [1].

Multiprojekt v PCS 7 je obvykle tvorený niekoľkými spojenými projektmi, vždy minimálne projektom AS a OS. Projekty AS sú automatizačné systémy, takže jednotlivé riadiace PLC rady S7-400. OS sú projekty operátorského systému. Multiprojekt je vytváraný, upravovaný a archivovaný z prostredia Simatic Manager.



Obr. 1.1 - Príklad riadiaceho systému postaveného na PCS 7 [2]

1.1.1 Automatizačný systém AS

Continuous function chart (CFC)

CFC editor je základný nástroj pri tvorbe AS. Je to grafický editor kontinuálnych automatizačných funkcií, v ktorom sa umiestňujú a logicky prepájajú predvytvorené systémové bloky. Editor podporuje autorouting a integruje konfiguráciu HMI správ. Špeciálne možnosti ako chart-in-chart, viacnásobné použitie chartu ako bloku alebo SFC type bloku vo forme inštancií umožňujú úspornejšiu a prehľadnejšiu tvorbu programu.

Sequential function chart (SFC)

SFC editor je určený na grafickú tvorbu sekvenčných automatizačných funkcií. Obvykle sa používa na riadenie základných funkcií vytvorených v CFC. SFC môžu byť v PCS 7 dva druhy - SFC plan alebo SFC type.

SFC plan

SFC plan je výhodný pre funkciu použitú jedenkrát. Každý SFC plan má štandardné vstupy a výstupy pre sledovanie stavu a riadenie sekvencie programom alebo užívateľom. SFC plan sa ako aj iné bloky môže umiestniť ako blok do CFC. Jedno SFC môže obsahovať až 8 nezávislých sekvencií, využiteľných napríklad pre stavy sekvencie ako Holding, Aborting alebo pre rôzne operačné stavy.

SFC type

SFC type sú štandardizované sekvencie, ktoré môžu byť použité viacnásobne. Pracuje sa s nimi podobne ako s normálnymi funkčnými blokmi, takže ich je možné umiestniť ako inštanciu do CFC a prepojiť s inými blokmi. Zmeny v origináli sa automaticky prejavia vo všetkých inštanciách. SFC type môže obsahovať až 32 sekvencií.

Siemens STEP 7

CFC a SFC sú grafické nadstavby nad systémom STEP 7, a ich kompiláciou sa tvoria štandardné bloky systému STEP 7. Tak isto je možné do chartu vložiť vlastné funkčné bloky (FB) alebo funkcie (FC). Táto možnosť je využívaná hlavne pri tvorbe blokov pre komunikáciu so zariadeniami iných výrobcov, doplnenie chýbajúcich funkcií alebo tvorbu špeciálnych blokov zložitejšieho riadenia.

1.1.2 Operátorský systém OS

Operátorský systém v PCS 7 umožňuje jednoduché a bezpečné riadenie procesu. Operátor môže sledovať stav procesu v rôznych pohľadoch a v prípade potreby vykonať zásahy a úpravy. Architektúra OS je flexibilná a dá sa prispôsobiť špecifickým podmienkam jednotlivých výrobných podnikov a požiadavkám zákazníka. Podľa požiadaviek je možné systém vybaviť jednou operátorskou stanicou (OS single stations) alebo viacužívateľskou architektúrou klient/server. Každá stanica môže mať až 4 monitory, vďaka čomu umožňuje pohodlné riadenie niekoľkých častí technologického procesu súčasne na jednej stanici. Vo väčších prevádzkach je vhodné použitie samostatného Central Archive Serveru CAS, čím sa zníži zaťaženie OS Serverov. OS obsahuje pre jednotlivé stupne hierarchie technologické obrazovky, na ktoré sa pri kompilácii vygenerujú ikony objektov z príslušnej hierarchie z projektu AS. Toto veľmi zjednodušuje a urýchľuje tvorbu vizualizácie, pretože systém pri kompilácii vytvorí nielen ikony, ale aj všetky prepojenia potrebné pre správne previazanie objektu v AS a OS.

1.2 CEMAT

CEMAT je riadiaci systém špecificky navrhnutý pre cementárne a je v tomto odvetví široko akceptovaný. Po celú dobu jeho existencie bol vyvíjaný v úzkej spolupráci s mnohými výrobcami cementu po celom svete. Aktuálna verzia CEMAT 7 je založená na mainstreamovom riadiacom systéme Siemens SIMATIC PCS 7, vďaka čomu ponúka unikátnu, otvorenú architektúru pre moderné, rozšíriteľné a ekonomické riešenia pre cementárenský priemysel. CEMAT umožňuje využitie všetkých vlastností a funkcií SIMATIC PCS 7 a pridáva ďalšie možnosti ako diagnostiku porúch, špeciálne funkčné bloky a blokácie vyžadované vo výrobe cementu.

CEMAT obsahuje množstvo modulov, z ktorých najpoužívanejšie sú:

C_GROUP - Skupina je nadradený modul pre spúšťanie, zastavovanie a monitoring technologicky zoskupených sekcií. Modul umožňuje vizualizáciu stavu a detailnú diagnostiku chýb v tejto sekcii závodu.

C_ROUTE - Cesta je modul pre výber dopravných smerov v rámci skupiny. V závislosti na zvolenej ceste je možné spúšťať iba určité pohony patriace do skupiny, vo vizualizácii stavu sa prejavia iba objekty patriace pod zvolenú cestu. Modul umožňuje prevolenie ciest bez zastavenia toku materiálu.

C_SELECT - Navolenie môže byť použité pre akúkoľvek funkciu navolenia, na rozdiel od cesty neumožňuje detailnú analýzu porúch, je ale jednoduchšie na použitie.

C_DRV_1D - Modul na riadenie, monitoring a vizualizáciu jednosmerných pohonov.

C_DRV_2D - Modul na riadenie, monitoring a vizualizáciu obojsmerných pohonov.

C_VALVE - Modul na riadenie, monitoring a vizualizáciu ventilov.

C_DAMPER - Modul na riadenie, monitoring a vizualizáciu klapiek.

C_MEASURE - Modul merania sa používa na spracovanie analógových hodnôt z S7 periférnych kariet alebo na zobrazenie existujúcich fyzických hodnôt. Modul obsahuje funkcie ako potlačenie špičiek, monitoring prekročenia limít a gradientu. Vo vizualizácii je možné sledovať trend hodnoty

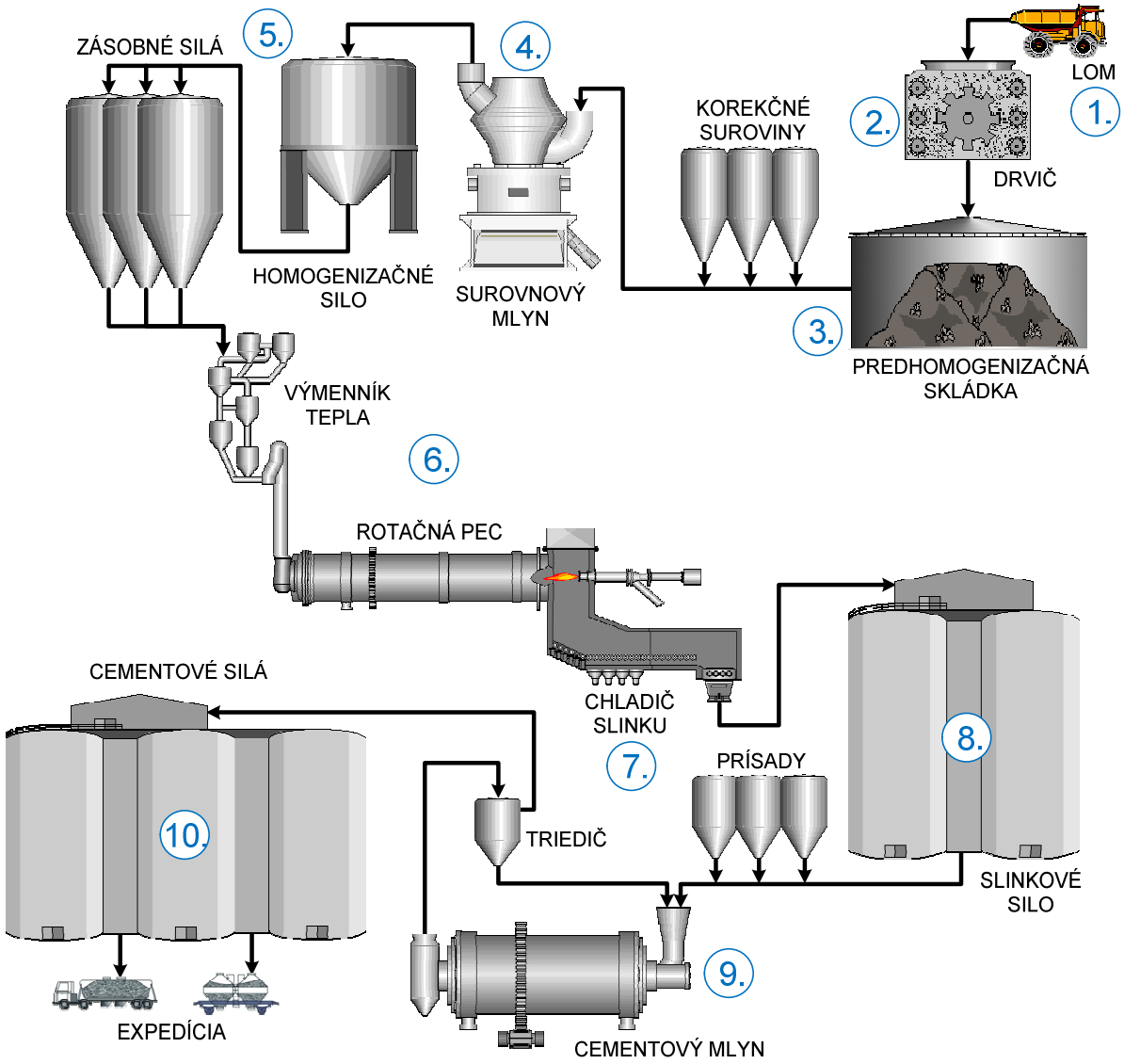
C_ANNUNC - Hlásenie sa používa na zobrazenie stavu signálu, prípadne zobrazenie alarmu.

C_ANNUN8 - Tento modul umožňuje zobraziť 7 individuálnych alarmov (8. je spoločný).

C_INTERL - Blok vyžívaný na implementáciu zobrazenia blokácií v iných moduloch

C_PID3 - Obsahuje v sebe štandardný blok PID, pridáva mu však ďalšie možnosti, hlavne možnosť prepínania medzi troma sadami parametrov.

1.3 Postup výroby cementu



Obr. 1.2 - Schéma postupu výroby cementu

1. Ťažba surovín

Hlavnou surovinou na výrobu cementu sú vápenaté horniny. Tie sa ťažia povrchovým spôsobom v lomoch, obvykle v blízkosti cementárne. Tvrdá hornina je oddeľovaná odstrelmi, následne je odoberaná nakladačmi a dopravovaná (autami, pásovými dopravníkmi) do drviarne. Pri ťažbe suroviny sú vykonávané pravidelné analýzy chemického zloženia ťažených hornín s cieľom pripraviť optimálnu zmes pre výrobu surovinovej múčky.

2. Drvenie surovín

Drvenie suroviny prebieha podľa charakteru horniny v čeľusťových, kladivových alebo odrazových drvičoch. Zrnitosť výstupného materiálu sa dá v určitom rozsahu meniť, v sústave drvičov bývajú často vložené triediče k vráteniu nadlimitných zvyškov.

3. Predhomogenizácia surovinovej zmesi

Rozdrvená surovina putuje do surovinového skladu alebo na predhomogenizačnú skládku. Základným predpokladom dosiahnutia stálej a vysokej kvality slinku a cementu je vysoký stupeň homogenity vstupnej surovinovej zmesi. Pretože kvalita a zloženie vstupných surovín značne kolíše, používa sa pri výrobe obvykle niekoľko stupňov homogenizácie. Prvým stupňom je predhomogenizačná skládka, ktorá zároveň slúži ako zásobáreň suroviny. Požadovaná homogenita je zabezpečená systémom zakladania a odoberania suroviny.

4. Mletie surovinovej zmesi

Mletie patrí k najdôležitejším fázam prípravy surovín pred výpalom. Počas mletia je rozdrvená a homogenizovaná surovina mletá na surovinovú múčku vhodnú pre výpal v peci. Jemnosť mletia má rozhodujúci význam na priebeh procesu slinovania a rýchlosti tvorby slinku pri výpale, preto aj táto časť obyčajne obsahuje vzorkovač.

5. Homogenizácia suroviny

Rozdrvená a rozomletá surovina postupuje do homogenizačného sila, a následne do zásobných síl, kde je tiež možné odoberať vzorky suroviny. Vďaka tomu je ešte možné podľa výsledkov analýz z laboratória upraviť zloženie suroviny.

6. Výpal slinku

Výpal slinku je najdôležitejší úsek technologického postupu pri výrobe cementu. Surovinová múčka zo síl prechádza výmenníkom, kde dochádza k predhriatiu suroviny a jej čiastočnej dekarbonizácii. Najnovším trendom je využitie kalcinátorov medzi výmenníkom a samotnou pecou, čo zvyšuje efektivitu výpalu a výkon pece. Pálením surovinovej múčky až na medz slinutia (cca 1400 °C) vznikajú umelé, tzv. slinkové minerály. Slinok sa vypaľuje v cementárenských peciach, v súčasnosti najpoužívanejšie sú rotačné pece. Sú to v podstate oceľové valce vyložené žiaruvzdornou výplňou. Pece bývajú dlhé od 60 - 130 metrov s priemerom 3 - 7 metrov. Pec je uložená s pozdĺžnym sklonom 3 - 7 stupňov.

Ako palivo sa v súčasnej dobe stále viac používajú alternatívne palivá, ktoré nahradzujú primárne fosílne palivá (uhlie, zemný plyn). Alternatívnymi palivami sú napr. drvené pneumatiky, upravené spáliteľné zložky komunálneho odpadu, použité oleje, mäsokostná múčka apod.

7. Chladenie slinku

Slinok vypálený v peci sa ochladzuje v chladiči slinku, najčastejšie roštových alebo planétových. Toto prudké ochladenie stabilizuje vytvorené slinkové minerály a vznikne portlandský slinok.

8. Slinkové silo

Ochladený slinok sa následne uskladňuje vo veľkopriestorových zásobníkoch. Tie slúžia na odležanie slinku, jeho dodatočné vychladnutie a tiež ako dostatočná zásoba slinku pre následnú výrobu.

9. Mletie cementu

Po odležaní sa portlandský slinok mele spolu s ďalšími prísadami, napr. sadrovec, struska a popolček. Jemnosť mletia je zásadnou výrobnou operáciou k použitiu cementu, jemne mleté cementy majú väčšiu pevnosť. Na mletie cementu sa najčastejšie používajú guľové mlyny, čo sú v podstate duté valce otáčajúce sa okolo vodorovnej osi. Vo vnútri mlyna sú železné gule, ktoré sú pôsobením odstredivých síl vynášané nahor, odkiaľ po dosiahnutí určitej výšky padajú . Materiál je padajúcimi a prevaľujúcimi sa guľami postupne rozomieľaný. Výstupný materiál z mlyna ešte prechádza triedičom, ktorý odlúči prášok s požadovanou jemnosťou a zvyšok (spätnú krupicu) vráti do mlyna. Výstup z mlyna sa tiež priebežne kontroluje v laboratóriu a to na chemické zloženie aj granulometriu.

10. Cementové silá a expedícia

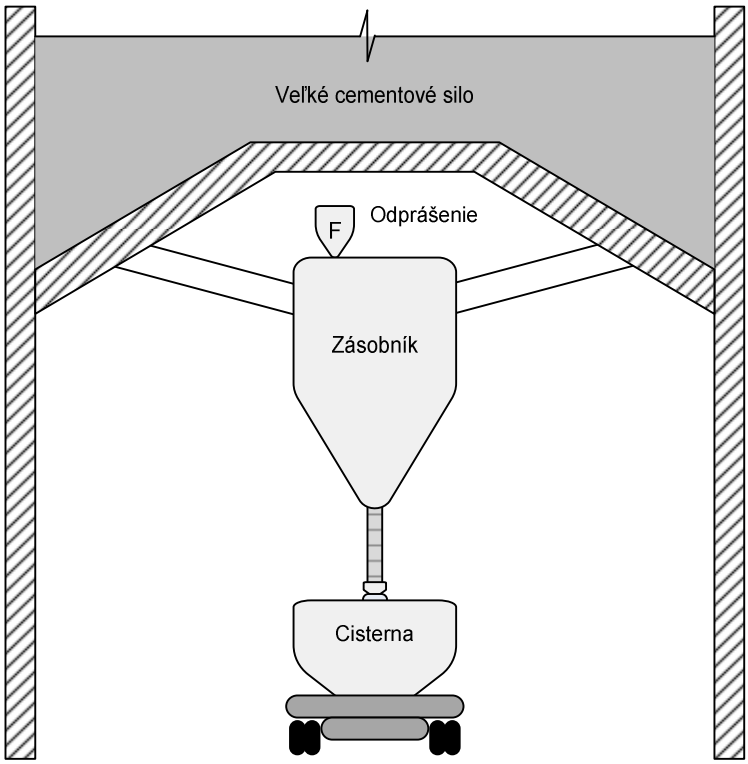
Pomletý cement sa pásovými dopravníkmi podľa jeho druhu dopraví do príslušného cementového sila, kde sa uskladňuje a potom expeduje. Z cementových síl sa pravidelne odoberajú vzorky, pre kontrolu kvality výstupného produktu. Expedícia obvykle prebieha autocisternami alebo špeciálnymi železničnými vagónmi, prípadne môže obsahovať balička do vriec a paletovač. [3]

2. Projekt Astana - Expedícia

2.1 Rozbor problému

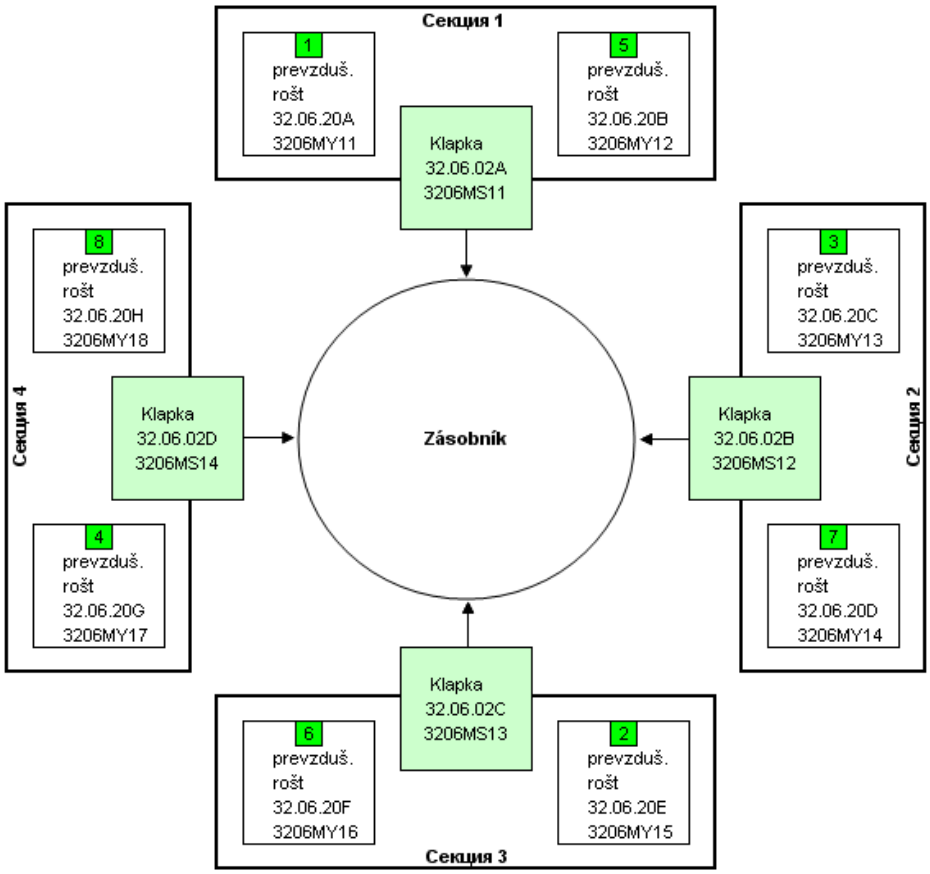
Expedícia sa skladá zo štyroch veľkých síl, ktoré sú technologicky rovnaké, takže ďalší popis pre silo 1 bude platiť pre všetky. Každé silo je technologicky rozdelené do troch skupín:

80G11 - Preprava cementu z veľkého sila do zásobníku
80G12 - Plnenie cementu zo zásobníku do cisterny
80G13 - Odprášenie



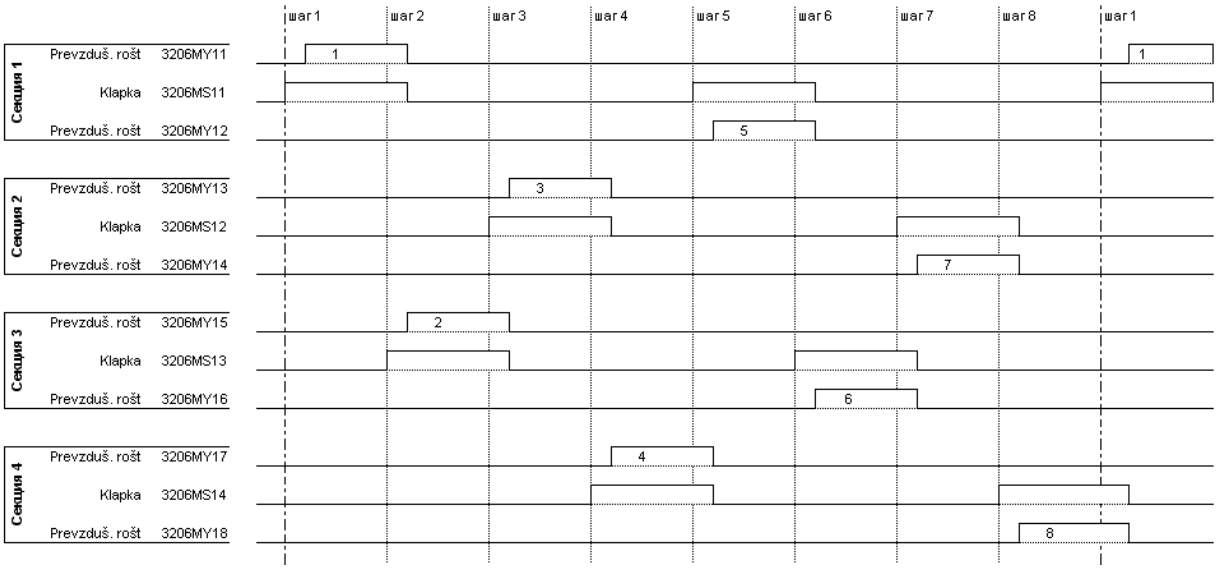
Obr. 2.1 - Princíp fungovania expedície

Plnenie sila patrí pod časť Cement Transport, ktorá je súčasťou PLC riadiacich cementové mlyny, Expedícia riadi vyprázdňovanie sila. Pri požiadavku na plnenie cisterny sa časť cementu najprv z veľkého skladovacieho sila premiestni do menšieho zásobníku, odkiaľ sa potom plní do cisterny. Aby sa cement z veľkého sila odoberal rovnomerne a nezostal niekde vytvorený previs, odoberá sa cez štyri otvory pomocou žľabov. Ku každému žľabu patrí jedna klapka a dva prevzdušňovacie ventily. Klapka slúži na ovládanie prístupu do žľabu, cez ventily je tlačený vzduch do prevzdušňovacích roštov. Vtláčaním vzduchu cez tieto rošty sa cement nadvihne, prevzdušní a získa vlastnosti tekutiny, vďaka čomu cez otvorenú klapku môže "vytiecť" do žľabu ktorým je prepravovaný ďalej do zásobníku. Schéma priradenia klapiek a ventilov je na obr. 2.2.



Obr. 2.2 - Schéma umiestnenia klapiek a prevzdušňovacích roštov (pohľad zhora)

Pre túto funkciu je navrhnutá sekvencia otvárania jednotlivých klapiek a ventilov, ktorá zabezpečí správne odoberanie cementu z veľkého sila (obr. 2.3). Táto sekvencia bola daná vo funkčných špecifikáciách dodaných zákazníkom.



Obr. 2.3 - Sekvencia otvárania klapiek a ventilov

Ďalšie dané požiadavky na riadenie tejto časti:

- Pri poruche klapky alebo ventilu preskočiť aktuálnu fázu
- Pri tretej poruche tej istej klapky (ventilu) odstaviť plnenie
- Možnosť meniť operátorom čas fázy a čas prekrytia (Overlap Time)
- Zobrazovať vo vizualizácii číslo aktuálnej fázy a zostávajúci čas do nasledujúcej fázy

Z týchto požiadaviek vyplýva, že musíme počítať chyby pre každú klapku a ventil, v SFC však toto nie je možné, takže sme využili externé čítače. Na zobrazovanie a zadávanie času v PCS 7 nie je k dispozícii žiadny blok, preto sme ho museli vytvoriť.

Ďalším problémom sú samotné klapky na žľaboch. Pri plnení zásobníku sa sleduje hladina cementu a pri prekročení nastaveného limitu sa klapky budú otvárať iba čiastočne (poloha middle). CEMAT obsahuje objekt pre klapku, avšak tá je buď dvojpolohová (opened/closed), alebo s pozicionérom (0-100%). V tomto prípade sú však použité klapky trojpolohové (opened/middle/closed), museli sme ich teda upraviť.

Aby bolo možné spustiť plnenie cisterny, je nutné aby bola plniaca hubica dosadnutá na otvore cisterny. To musí zabezpečiť obsluha (vodič) pomocou tlačidiel na ovládacom pulte, pretože k tomu treba vidieť na cisternu. Pohyb je možný pozdĺžne a vertikálne, správne dosadnutie je snímané čidlom na hubici. Po dosadnutí hubice a dosiahnutí definovanej hladiny v zásobníku je možné spustiť samotné plnenie, ktoré má dve fázy. V prvej je klapka plniaceho žľabu otvorená naplno, po dosiahnutí definovanej hladiny v cisterne sa klapka privrie a plní sa menšou rýchlosťou. Po ukončení plnenia sa zvyšný cement z plniacej hubice a žľabu odsaje a cez filter vráti do zásobníku.

Odprášenie musí byť aktívne počas celej doby behu plnenia, musí to byť teda jedna z podmienok spustenia ostatných skupín. Samotné odprášenie sa zapína postupne, najprv ventilátor, filter a následne turniket. Počas behu sa filter v daných cykloch regeneruje, pri vypnutí odprášenia je nutné vykonať ešte jeden cyklus regenerácie. Cementový prach zachytený na filtri sa turniketom dostáva do zásobníku.

2.2 Realizácia

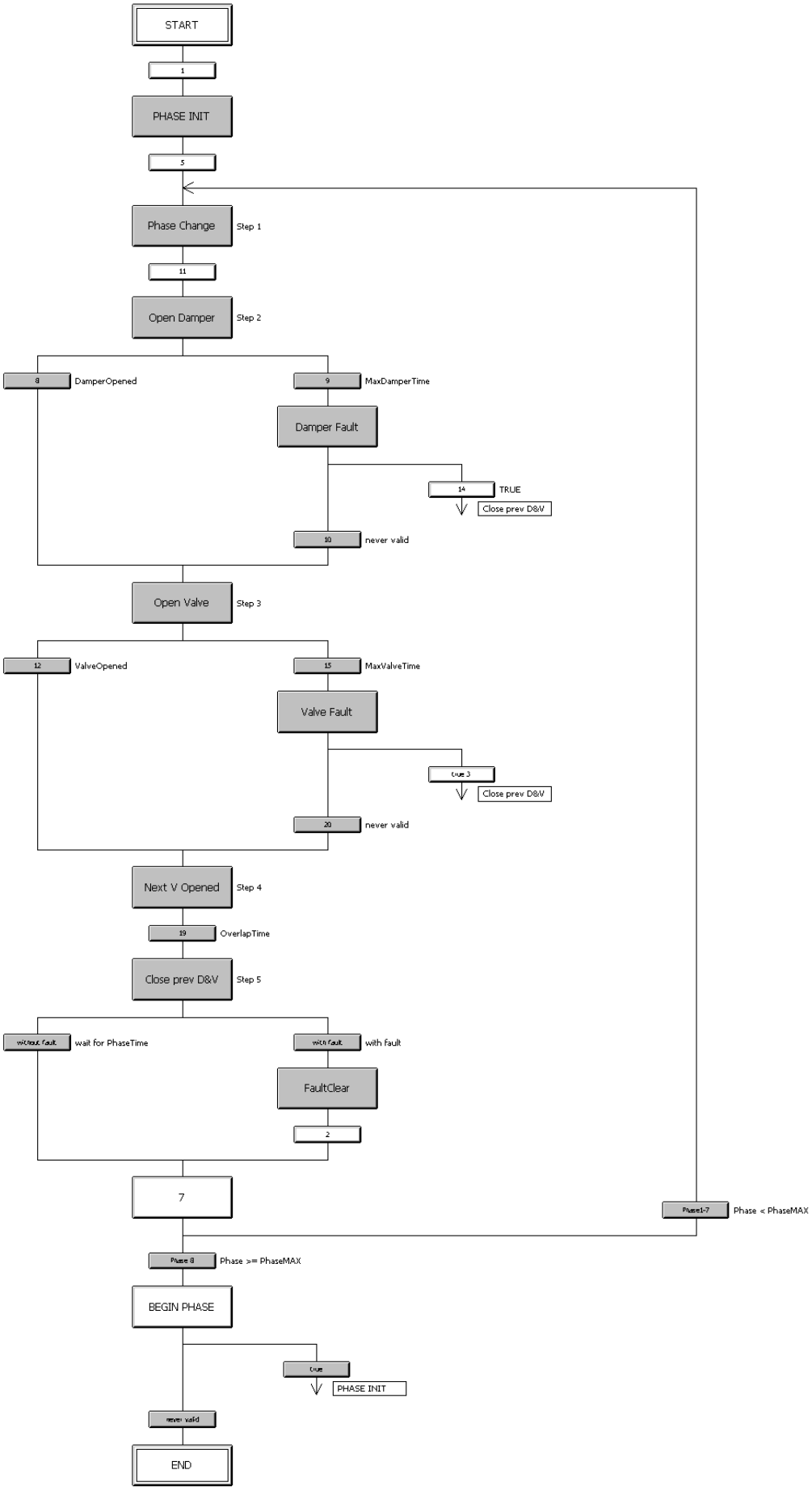
2.2.1 Sekvencia plnenia zásobníku

Podľa obr. 2.3 má celá sekvencia 8 fáz (krokov), v skutočnosti ale každá jedna fáza obsahuje postupnosť niekoľkých ďalších stavov:

1. Otvorenie klapky
2. Kontrola otvorenia klapky (inak chyba)
3. Otvorenie ventilu
4. Kontrola otvorenia ventilu (inak chyba)
5. Čakanie na Overlap Time
6. Zatvorenie klapky a ventilu z predchádzajúcej fázy
7. Čakanie na uplynutie Phase Time
8. Prechod na ďalšiu fázu

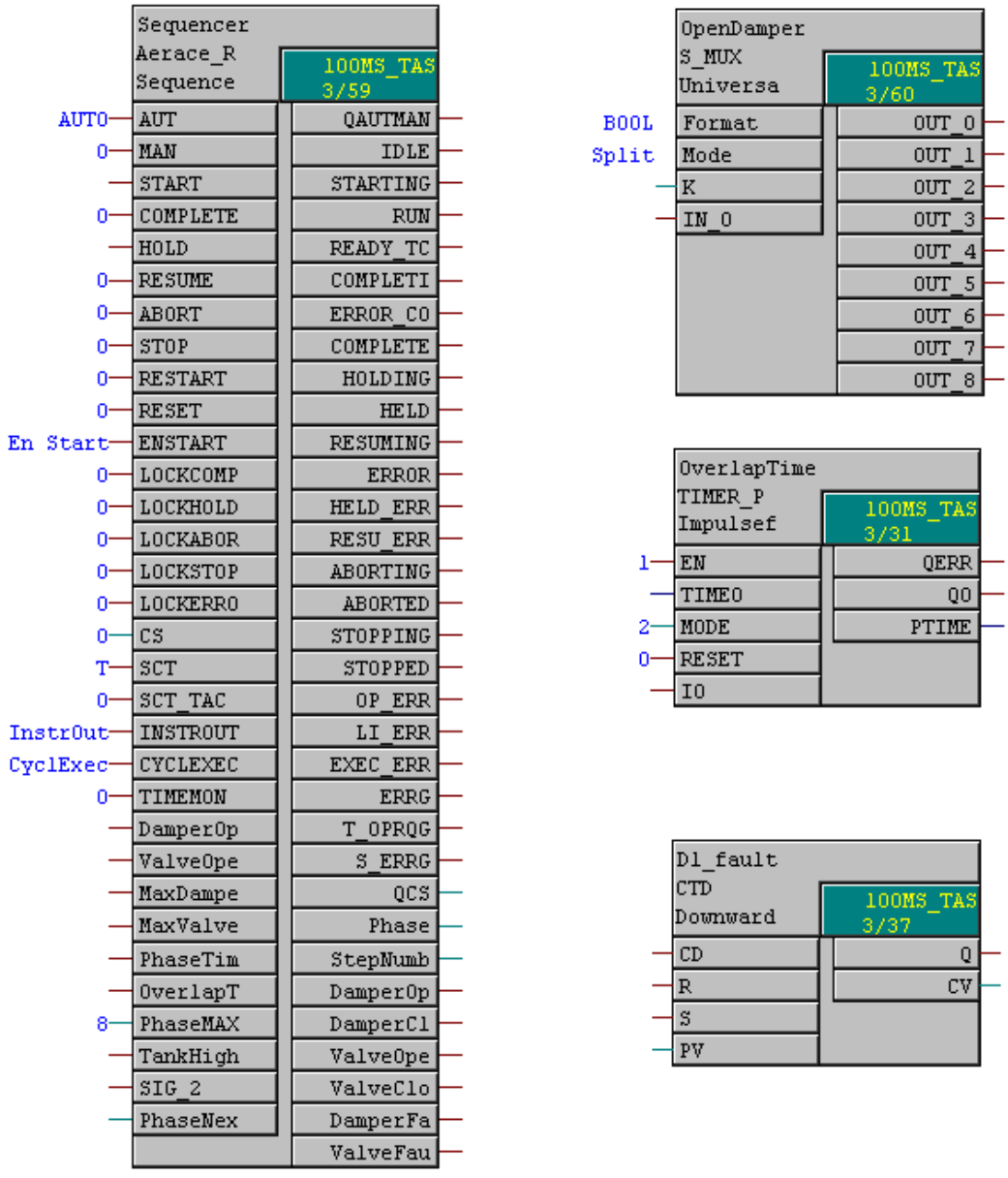
Sekvencer zložený z ôsmych takýchto fáz by bol značne rozsiahly a neflexibilný. Ako najlepšie riešenie sme teda navrhli použitie sekvenceru na riadenie jednej fázy ktorá by sa opakovala pre jeden cyklus osemkrát. V každom kroku sa riadia iné klapky a ventily, takže ich povely na otvorenie a zatvorenie sú multiplexované podľa čísla fázy. Na tento účel sme využili blok S_MUX, ktorý sme mali vo firme vytvorený v inom projekte. Je to univerzálny MUX/DEMUX blok pracujúci s rôznymi dátovými typmi.

Sekvencia ktorú sme naprogramovali v SFC (obr. 2.4) sa teda zopakuje pre každú fázu, pričom výstup Phase značí číslo aktuálnej fázy. V kroku OpenDamper sa vyšle povel na otvorenie príslušnej klapky. Pokiaľ sa klapka otvorí, sekvencia pokračuje ďalej otvorením ventilu. Ak sa však do definovaného času klapka neotvorí, program skočí do kroku DamperFault, a potom na koniec fázy (Close prev D&V). Detekcia chyby pre ventil a klapku je rovnaká, ak sa detekuje chyba klapky, príslušný ventil sa neotvára, preskakuje sa celý krok. Počet chýb každej klapky a ventilu je počítaný čítačom v externej logike, keď na niektorej klapke alebo ventile nastane definovaný počet chýb, sekvencia sa zastaví (stav Hold) a operátorovi sa zobrazí hlášku o chybe. Ovládanie povelov klapiek a ventilov je riešené multiplexom podľa čísla fázy. Povel na otvorenie klapky (ventilu) je daný v aktuálnej fáze, povel na zatvorenie je daný v nasledujúcej fáze. Je to zrejmé zo zadanej sekvencie, pretože klapka a ventil z predchádzajúcej fázy sa zatvárajú až po otvorení ďalšej klapky a ventilu, a uplynutí OverlapTime. Aby bolo možné meniť čas fázy aj čas prekrytia sú na to použité externé časovače.



Obr. 2.4 - Sekvencer

Bloky použité na zabezpečenie externých funkcií k sekvenceru (obr. 2.5) sú okrem bloku S_MUX štandardné PCS 7 bloky. Blok S_MUX bol vytvorený našou firmou pre potreby starších projektov, je však možné ho využívať v akomkoľvek inom. Jeho funkcia je univerzálna, dokáže pracovať ako multiplexer aj demultiplexer (podľa parametra Mode) pre rôzne dátové typy (parameter Format).

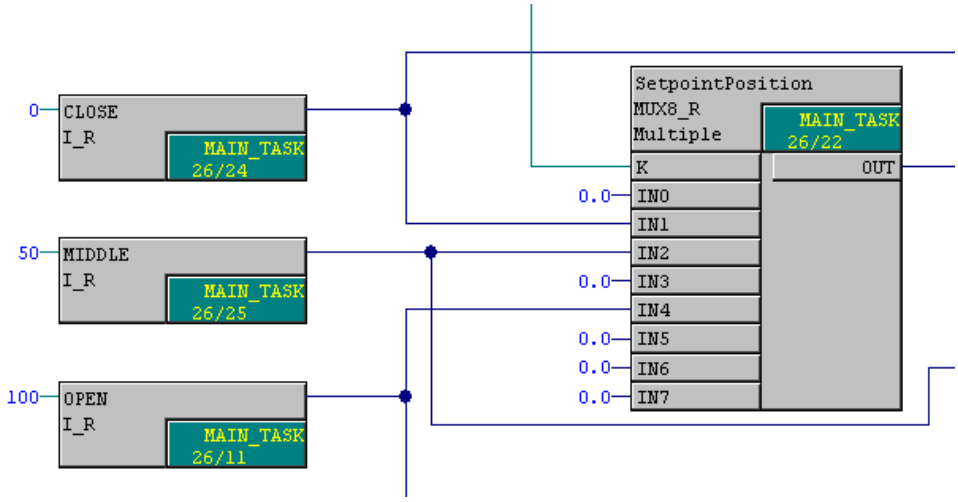


Obr. 2.5 - Bloky v CFC (Sekvencer, Multiplexer, Timer, Counter)

V prípade prepínania povelov je teda Format nastavený na BOOL a Mode na Split. Binárna hodnota zo vstupu IN_0 sa potom prepne na príslušný výstup OUT_X podľa vstupu K.

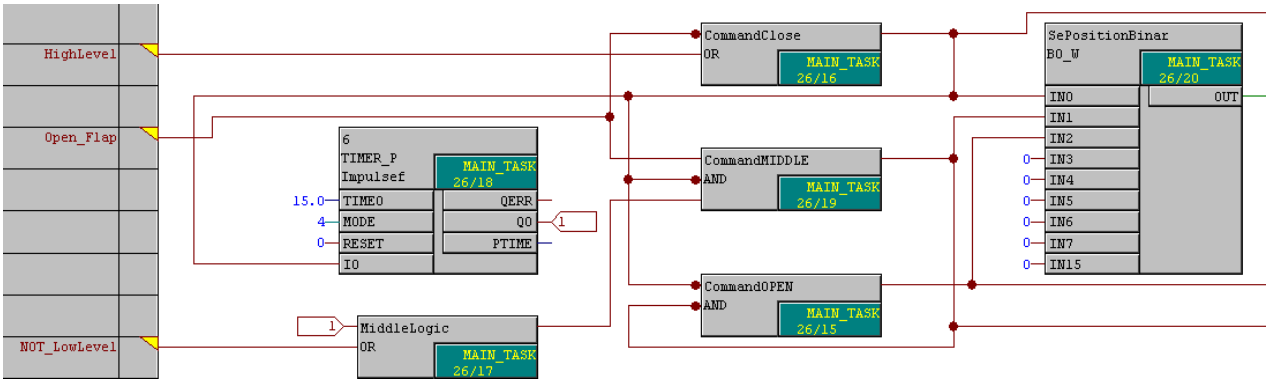
2.2.2 Trojpolohová klapka

Riadenie trojpolohovej klapky sme vyriešili s využitím módu s pozicionérom. V tomto móde sa klapka neriadi binárnymi príkazmi Open/Close, ale spojitou hodnotou Setpoint, čiže požadovaným otvorením. Klapka samozrejme musí mať informáciu o aktuálnej polohe. To je v tomto prípade problém, pretože my máme informáciu iba o troch polohách. Na vyriešenie tohto problému sme znova použili blok multiplexeru (obr. 2.6), ktorým sa prepínajú na Setpoint klapky hodnoty 0, 50 alebo 100 percent, podľa žiadanej polohy klapky.



Obr. 2.6 - Multiplexovanie Setpointu

Hodnotu na výstupe riadi výstup bloku prevodníku BO_W, do ktorého sú zapojené signály Open, Middle a Close (obr. 2.7). Tieto signály sú získané logickou kombináciou povelu na otvorenie klapky a signálov High a Low limit z merania výšky hladiny príslušného zásobníku.



Obr. 2.7 - Logika vytvorenia príkazov pre klapku

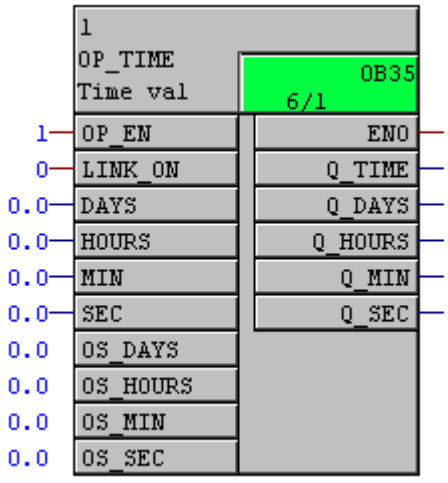
Nastavenie hodnoty aktuálnej pozície je vytvorené obdobne, akurát na blok BO_W sú pripojené priamo signály koncových spínačov klapky.

2.2.3 Zobrazenie a zadávanie času

Pretože samotný PCS 7 ani CEMAT neobsahujú blok pre zadávanie a zobrazovanie času, museli sme tento blok vytvoriť. Kompletný blok pozostáva z troch častí. Prvou je funkčný blok (FB), ktorý vykonáva samotnú logiku vstupu a spracovania času. Druhou časťou je ikona, ktorá sa vygeneruje na vizualizačnú obrazovku, treťou je tzv. faceplate, čo je okno, ktoré sa otvorí po kliknutí na ikonu.

Blok OP_TIME sme sa snažili navrhnúť čo najuniverzálnejšie s ohľadom na potreby aj v iných častiach projektu, prípadne aj iných projektoch, preto môže fungovať v dvoch módoch podľa stavu vstupu LINK_ON. Ak je vstup nastavený na nulu, blok ako vstup používa hodnoty zadané operátorom z vizualizácie. Tieto hodnoty sú do bloku naviazané cez IN/OUT piny s prefixom OS. V prípade stavu LINK_ON = 1 sa zadávanie vstupov z vizualizácie zablokuje a blok ako vstup používa vstupy DAYS, HOURS, MIN a SEC priamo z logiky. Všetky tieto vstupy sú typu Real.

Výstup Q_TIME slúži na pripojenie k vstupu časovačov, jeho hodnota je prepočítaný súčet sekúnd zo všetkých aktívnych vstupov bloku (podľa stavu LINK_ON). Výstupy Q_HOURS, Q_MIN a Q_SEC sú tak isto typu Real a slúžia hlavne pre potreby vizualizácie. Nie je však problém ich využiť aj inak. Blok je na obr. 2.8.



Obr. 2.8 - Blok OP_TIME

Na začiatku bloku sa vyhodnocuje, či budú použité vstupy z logiky, alebo z vizualizácie.

```
      A     #OP_EN
      AN    #LINK_ON
      =     #QOP_EN
      A     #LINK_ON            // Zadavanie operatorom alebo zo vstupov
      JC    M002
```

V prípade že LINK_ON má hodnotu 0, na výpočet sa použijú hodnoty z vizualizácie, kde je možné zadať samostatne dni, hodiny, minúty a sekundy. Tieto hodnoty sú previazané s INOUT pinmi bloku určenými práve pre tento účel.

```
      L     #OS_DAYS            // Nacitanie dni z OS
      T     #Q_DAYS
      L     2.400000e+001
      *R                        // Prevod na dni
      L     #OS_HOURS           // Nacitanie hodin z OS
      T     #Q_HOURS
      +R                        // Pripocitanie k celkovej sume (ACC2)
      L     6.000000e+001
      *R                        // Prevod na minuty
      L     #OS_MIN             // Nacitanie minut z OS
      T     #Q_MIN
      +R                        // Pripocitanie k celkovej sume (ACC2)
      L     6.000000e+001
      *R                        // Prevod na sekundy
      L     #OS_SEC             // Nacitanie minut z OS
      T     #Q_SEC
      +R                        // Pripocitanie k celkovej sume (ACC2)
      T     #Q_TIME             // Zapisanie sumy sekund na vystup Q_TIME
      BEU                       // Ukoncenie bloku
```

Tento kód zabezpečuje prevod zadaného času operátorom z vizualizácie na výstupy. Hodnoty sú obmedzené vo WinCC, takže o to sa blok nemusí starať. V prípade stavu LINK_ON = 1 sa na výpočet použijú hodnoty na vstupoch bloku.

```
M002: L     2.400000e+001
      L     #DAYS               // Nacitanie dni
      *R                        // Prevod na hodiny
      L     #HOURS
      +R                        // Pripocitanie hodin k sume
      L     6.000000e+001
      *R                        // Prevod na minuty
      L     #MIN
      +R                        // Pripocitanie minut k sume
      L     6.000000e+001
      *R                        // Prevod na minuty
      L     #SEC
      +R                        // Pripocitanie k sume
      T     #Q_TIME             // Zapisanie sumy sekund na vystup Q_TIME
```

Vďaka tomuto kódu je blok schopný previesť vstupné hodnoty časov na sekundy, pričom počíta súčet všetkých vstupov a je zabezpečený správny prevod v prípade zadania hodnôt minút a sekúnd vyšších ako 60, hodín ako 24 aj desatinných miest (0.5 min prevedie na 30 s).

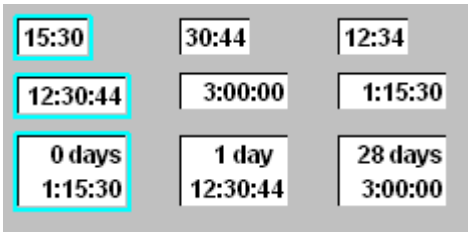
Celkovú sumu sekúnd na výstupe potrebujeme pre využitie v logike ako vstup pre časovače. Pre účely vizualizácie potrebujeme z tejto sumy vypočítať samostatne dni, hodiny, minúty a sekundy. Tento prepočet využíva ako zdroj predchádzajúcim kódom vypočítanú sumu sekúnd, a je nutný aby sa vo vizualizácii nezobrazovalo napríklad 90 s, ale 1m 30 s. V prvej časti kódu si do dočasných premenných modHours, modMin a modSec uložíme zvyšok po celočíselnom delení, Q_TIME, čo predstavuje jednotlivé počty sekúnd najprv bez celých dní, hodín a minút.

```
L     #Q_TIME
TRUNC
L     L#86400                    // Pocet sekund v jednom dni
MOD
T     #modHours                  // Ziskanie zvysneho poctu hodin
L     3600
MOD
T     #modMin                    // Ziskanie zvysneho poctu minut
L     60
MOD
T     #modSec                    // Ziskanie zvysneho poctu sekund
```

Počet celých dní, hodín a minút teda získame z jednotlivých čiastočných časov ich vydelením počtom sekúnd v daných časových intervaloch a ich orezaním o desatiné miesta príkazom TRUNC. Príkaz DTR prevedie DINT na REAL.

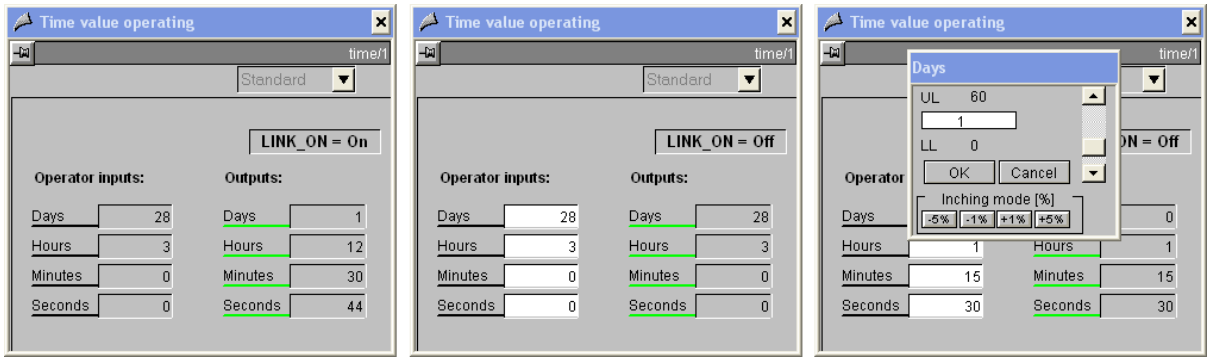
```
L     #Q_TIME
L     8.640000e+004
/R
TRUNC                            // Oddelenie celych dni
DTR
T     #Q_DAYS                    // Ulozenie dni

L     #modHours
L     3600                       // Oddelenie celych hodin
/D
DTR
T     #Q_HOURS                   // Ulozenie hodin

L     #modMin
L     60
/D                               // Oddelenie celych minut
DTR
T     #Q_MIN                     // Ulozenie minut

L     #modSec
DTR
T     #Q_SEC                     // Ulozenie sekund
```

Ikony pre OP_TIME sme vytvorili tri. V celom projekte sme totiž potrebovali zadávať rôzne časové rozsahy v rádoch sekúnd, minút a v jednom prípade dokonca 30 dní. Samozrejme je nevhodné zobrazovať ikonu s údajmi 0 dní, 0 hodín, keď maximálny čas je napríklad 5 minút. Na obr. 2.9 sú zobrazené rôzne stavy týchto troch typov ikon, modrý obdĺžnik značí, že tento objekt má otvorený faceplate (detail). Ako je vidieť na najnižšej ikone, mení sa aj nápis day/days podľa počtu dní, toto je dosiahnuté skriptom vo WinCC.



Obr. 2.9 - Ikony pre blok OP_TIME

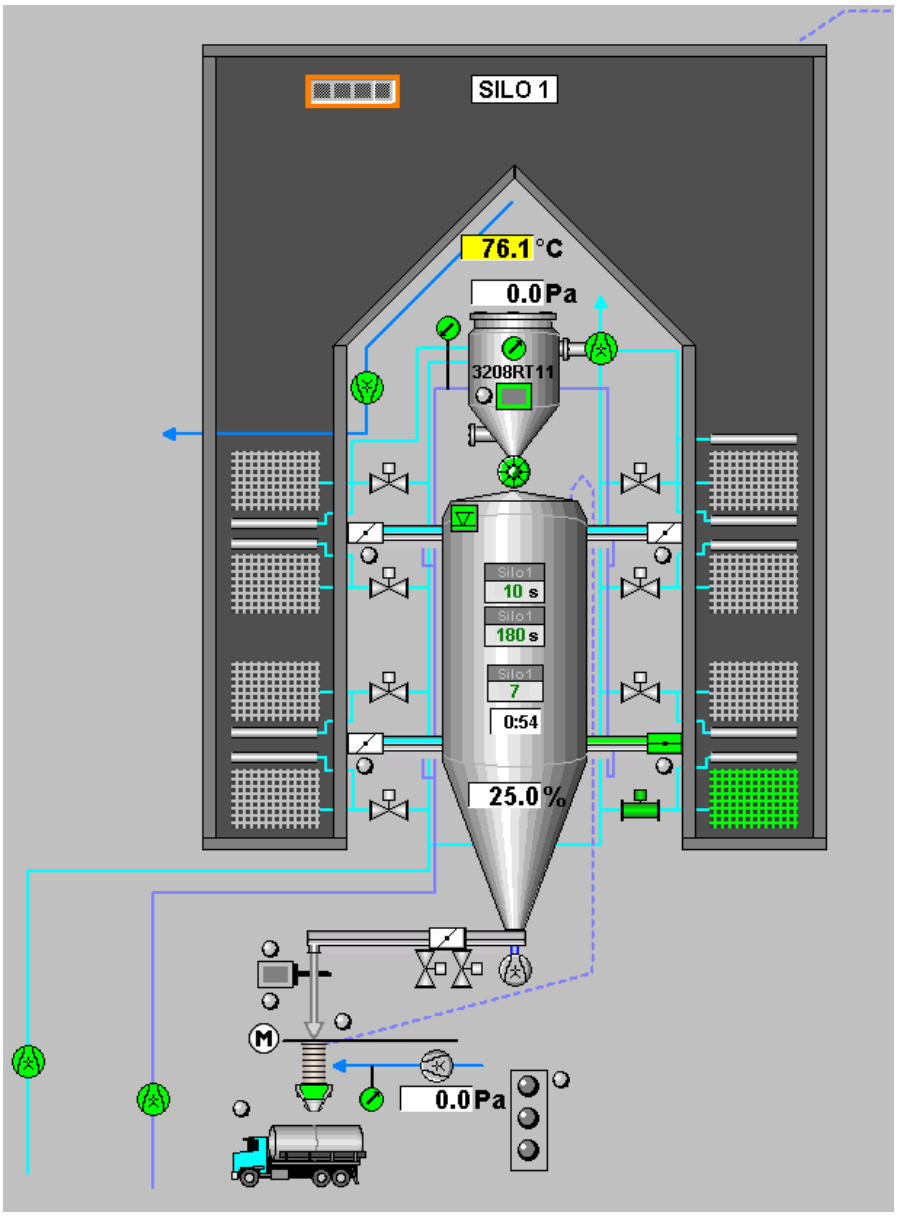
Detailnú obrazovku (Faceplate) sme navrhli čo najpodobnejšiu štandardným obrazovkám z PCS 7 Library. Na obr. 2.10 sú zobrazené rôzne stavy. V pravom hornom rohu je zobrazený stav vstupu LINK_ON, podľa ktorého operátor môže zistiť, odkiaľ sa čerpajú zobrazované hodnoty. Na prvom obrázku je vstup LINK_ON = 1, kedy sa hodnoty získavajú zo vstupov bloku a operátor ich nemôže meniť (použitie ako zobrazovač času), pôvodne zadané hodnoty sú však uložené (Operator inputs). Na ďalších obrázkoch je zadávanie povolené, na treťom je vidieť možnosti zmeny hodnoty - priame vpísanie, pomocou posuvníku alebo percentuálna zmena. Tento dialóg vychádza zo štandardného WinCC dialógu, ten je však určený na zadávanie reálnych čísel. Preto sme vytvorili tento nový, čo si vyžiadalo aj úpravu v Globálnych Scriptoch WinCC.



Obr. 2.10 - Faceplate pre OP_TIME v rôznych stavoch

2.2.4 Vizualizácia

Po vytvorení riadiacich funkcií podľa požadovaných špecifikácií s využitím vytvorených špeciálnych blokov sme vytvorili vizualizačné obrazovky, na ktorých sú zobrazené jednotlivé silá a vygenerované ikony technologických objektov. Výsledná podoba jedného sila je na obr. 2.11. Zobrazuje stav sila 1 pri spustenom plnení zásobníku, pričom sa nachádza vo fáze 7, ktorá skončí za 54s. Odsávanie a filter 3208RT11 je spustený a plnenie do cisterny ešte nezačalo. Na obr. je vidieť ikonu jednej cesty (oranžový obdĺžnik), ktorá je neaktívna, pretože je naviazaná na objekt patriaci do iného PLC, ktoré v tomto okamihu nebolo aktívne.

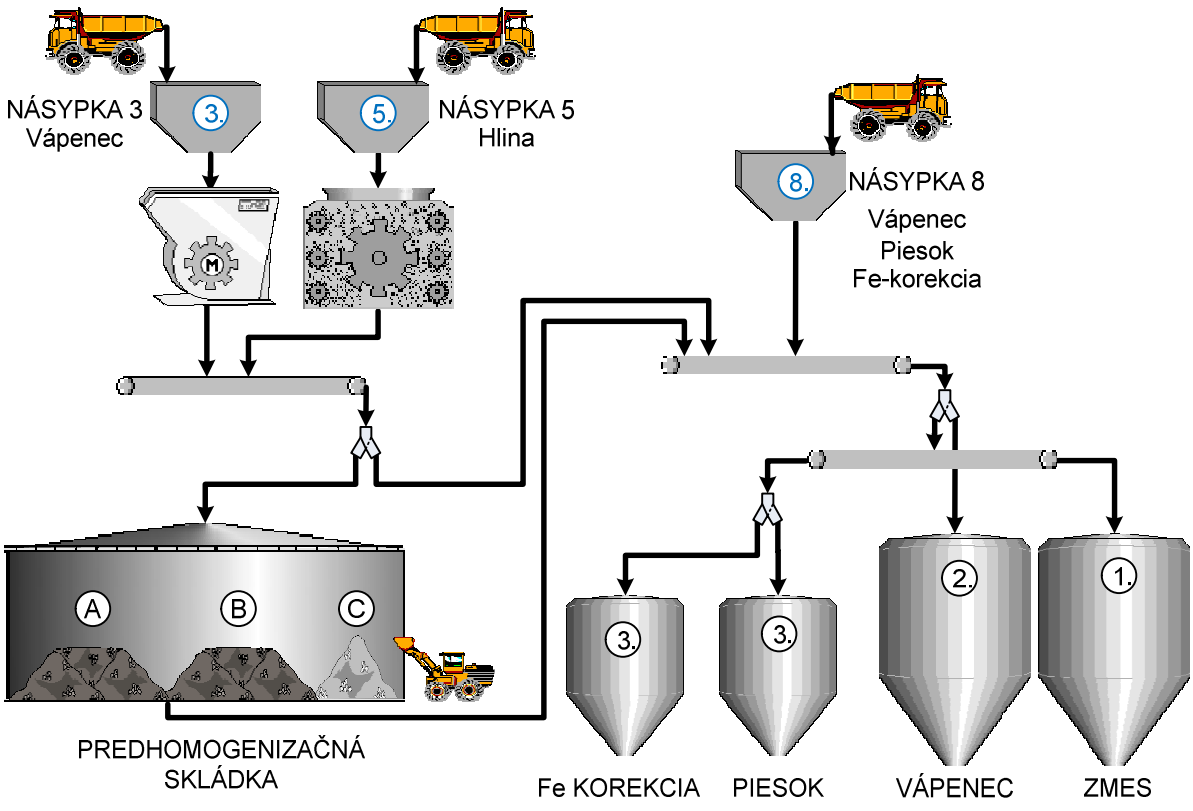


Obr. 2.11 - Vizualizácia prvého sila expedície

3. Projekt Astana - Drvenie a transport suroviny

Úlohou tejto časti technológie je príprava surovín pre surovinový mlyn. Suroviny vyťažené v lome sa pomocou nákladných automobilov privezú a vysypú do patričných násypiek, odkiaľ sa materiál postupne odoberá. Tento materiál sa transportuje na ďalšie spracovanie (drvenie, miešanie, homogenizáciu) a následne sa dopraví do skladovacích síl pred surovinovým mlynom.

Najdôležitejšou surovinou spracovávanou v tejto časti technológie je vápenec. Ten môže byť na základe analýz v laboratóriu prepravovaný buď čistý, alebo sa mieša s hlinou. Čistý vápenec sa dopravuje priamo z násypky 3 cez kladivový drvič do sila 2 alebo na skládku (halda C). Pri miešaní zmesi je zmiešavaný s hlinou, ktorá sa odoberá z násypky 5 a drví vo valcovom drviči. Táto zmes vápenca s hlinou sa musí dopraviť pomocou pásového zakladača na homogenizačnú skládku - halda A alebo B. Z tejto skládky sa zmes pomocou škrabáku zhomogenizuje a lopatovým hrabákom sa dostane na dopravnú cestu, ktorou je transportovaná do sila 1. Násypka 8 sa používa pre viac druhov materiálu - piesok, Fe-korekcia a čistý vápenec z haldy C. Z tejto haldy sa neodoberá škrabákom, ale nakladačom a k násypke sa vozí autami. Celá dopravná schéma toku materiálu je zobrazená na obr. 3.1.



Obr. 3.1 - Schéma drvenia a transportu suroviny

3.1 Rozbor problému

Hlavnou úlohou automatického riadiaceho systému pre túto technológiu je zabezpečiť transport materiálu zo zvoleného zdroja do správneho cieľa. Ďalšou nemenej dôležitou úlohou je správne miešanie a pomer zmesi, ako aj zabezpečenie chodu obidvoch drvičov súčasne. Pretože do násypiek sa materiál vozí nákladnými autami a neobsahujú žiadne čidlo zaplnenia ani meranie hladiny, kontrola dostatku materiálu na miešanie správneho pomeru zmesi je kontrolované až na výstupe drvičov podľa váhy hliny a celkovej zmesi.

Ďalším problémom je zabezpečiť správnu cestu materiálu, pričom podľa požiadaviek zákazníka je nutné cesty prepínať za behu čo najefektívnejším spôsobom. To znamená že pohony ktoré boli použité v starej ceste a budú použité aj v novej, sa pri prepínaní nezastavia. Pri prepnutí ciest sa musí najprv celá cesta vyčistiť od pôvodného materiálu, preto má každý pás určený čas dobehu, ktorý stačí na jeho vyprázdnenie.

Pri práci s materiálom je nutné zaobstarať odprášenie celej technológie, v tomto prípade to zabezpečuje 6 odprašovacích skupín. Tie sú tak isto spúšťané podľa použitej dopravnej cesty, ale dajú sa spustiť aj samostatne. Platí pre ne rovnaká požiadavka neprerušeného behu pri prepínaní ciest, pokiaľ sú použité aj v novo navolenej ceste.

Súčasťou tejto časti technológie je zakladač a škrabák, ktorých riadenie nemá plne pod kontrolou nami vytvorený nadradený riadiaci systém. Ich riadenie je zabezpečované samostatnými PLC dodanými výrobcom týchto zariadení, ktoré komunikujú s nadriadeným systémom, od ktorého dostávajú povely. Pre začlenenie týchto zariadení do systému a možnosť pracovať s nimi štandardnou cestou bolo potrebné vytvoriť pre každé komunikačný blok, ktorý by zabezpečil prevzatie a odoslanie povelov, ako aj sprostredkovanie prijatých dát do logiky v CFC.

Samozrejme je tak isto potrebné vytvoriť detailnú vizualizáciu všetkých objektov patriacich do tejto časti technológie, ako aj vizualizáciu jednotlivých materiálových ciest. Rozdelenie obrazoviek bolo dané zákazníkom na základe ich skúseností pri tvorbe predchádzajúcich výrobných závodov.

3.2 Realizácia

3.2.1 Komunikačný blok U_EXT_DEVICE

Niektoré časti technológie sú dodané spolu s vlastným riadiacim automatom, ktorý sa stará o samotnú činnosť daného zariadenia, ako celok je však riadený nadradeným riadiacim systémom, pre takýto systém sa používa anglické označenie "package" (balíček). Komunikácia medzi nadradeným systémom a package je zabezpečená pomocou štandardných komunikačných funkcií AG_SEND a AG_RECV. Aby však bolo možné tieto dáta využiť a prepájať s ostatnými časťami programu v CFC, bolo nutné vytvoriť blok, ktorý by zabezpečil prijatie dát a ich prenos na výstup bloku, ako aj odoslanie dát zo vstupov bloku. Pretože celý projekt Astana obsahuje niekoľko desiatok takýchto zariadení, bolo by výhodné vytvoriť tento blok univerzálne použiteľný pre všetky tieto zariadenia. Rozdiely medzi jednotlivými zariadeniami sú hlavne v rôznom počte bitových príkazov, statusov a číselných dát ktoré sa pri komunikácii používajú, ich spoločným znakom je však to, že ako prvá je vždy časť obsahujúca bity, až po nej nasleduje číselná časť. V tab. 3.1 je vidieť, že prvých 30 bytov dát má rôzny význam, vždy sú to však bitové signály. Po nich nasleduje niekoľko číselných hodnôt, v tomto prípade všetky DWORD.

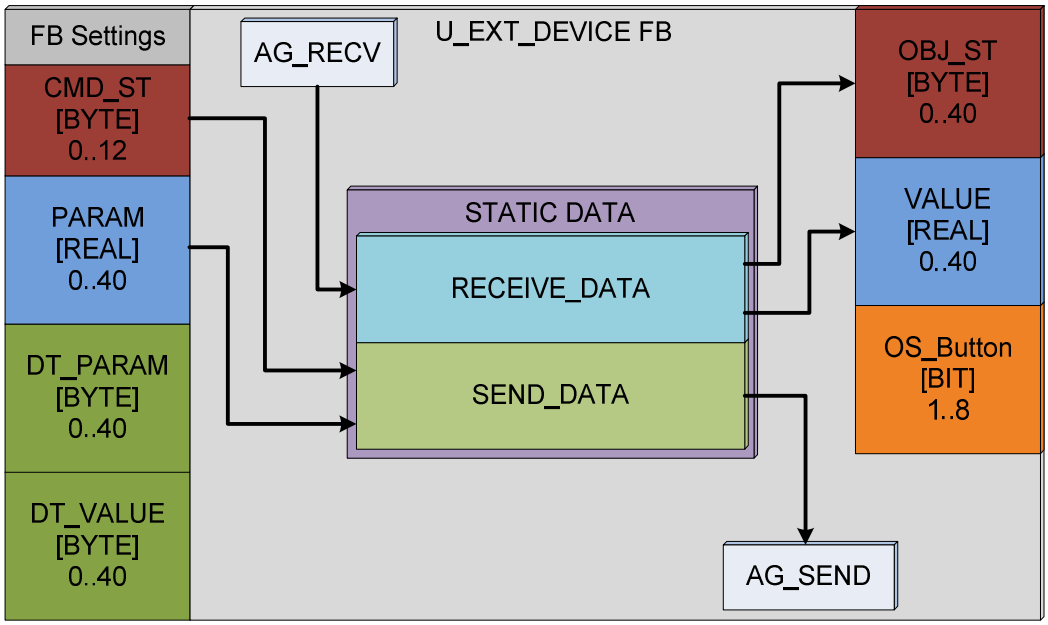
Address	Name	Data
00.0	0702RT01 Package Status	StatusByte
02.0	Faults 00 - 15	Faults bits
04.0	Binary sensors 00-15	Sensors bits
06.0	0702EH07	StatusByte
07.0	0702MA01	StatusByte
:	:	
22.0	0702EH10	StatusByte
23.0	0702PA01	StatusByte
24.0	0702MAxx_Faults	Faults bits
25.0	Reserve	
30.0	STACKER TRAVEL - ACTUAL SPEED SIDE 1	Analog status 00
34.0	STACKER TRAVEL - ACTUAL SPEED SIDE 2	Analog status 01
:	:	
54.0	HYDRAULIC STATION - OIL PRESSURE - GIB DOWN	Analog status 06
58.0	STACKER - ACTUAL INCLINATION	Analog status 07
59.0	Reserve	Analog status 08

Tab. 3.1 - Príklad prijatých dát od package 0702RT01

Štruktúra dát posielaných zariadeniu je rovnaká, rozdiel je vždy len v počte bytov prvej a druhej časti. Preto sme po analýze všetkých takýchto zariadení pre projekt Astana vymysleli štruktúru bloku (obr. 3.2), ktorá bude vyhovovať pre všetky zariadenia. Prvá skupina vstupov FB Settings slúži na nastavenie tohto FB, konkrétne obsahuje vstupy:

- ID_CONN - Connection ID pre komunikáciu s konkrétnym zariadením
- LADDR - Počiatočná adresa komunikačného modulu pripojeného k PLC
- I_CMD_BYTE - Počet bytov použitých pre bitové signály (posielané do zariadenia)
- I_PARAMS - Počet vstupných číselných parametrov
- Q_BYTES - Počet prijatých bytov zo zariadenia (bitových aj číselných dát)
- Q_ST_BYTES - Počet bytov použitých pre bitové signály (prijaté od zariadenia)

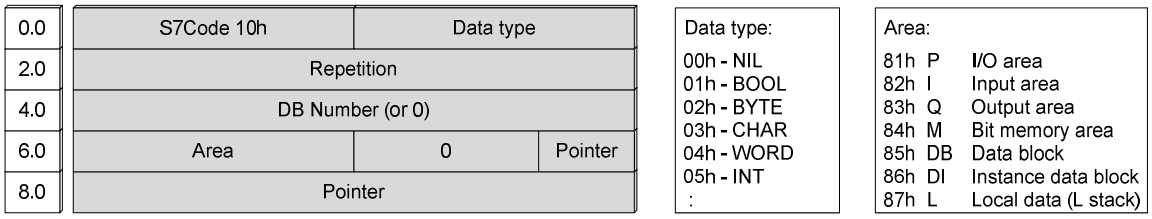
Podľa hodnôt na týchto vstupoch sa spracovávajú prijaté aj odosielané dáta. Druhá skupina sú bytové vstupy CMD_ST resp. výstupy OBJ_ST, ktoré obsahujú jednotlivé bitové signály. Nasledujúce vstupy PARAM a výstupy VALUE typu REAL slúžia na sprostredkovanie číselných dát. Typ REAL sme zvolili z dôvodu možnosti v tomto type uchovať aj celé čísla typu BYTE a INT. Práve na určenie pôvodného dátového typu slúžia vstupy DT_PARAM a DT_VALUE. Bitové výstupy OS_Button slúžia na pripojenie povelov tlačidiel z vizualizácie.



Obr. 3.2 - Diagram FB U_EXT_DEVICE

Aby bolo možné jednotlivé vstupy a výstupy vizualizovať, majú všetky nastavené potrebné parameter S7_m_c, a číselné vstupy a výstupy aj S7_shortcut na pridanie názvu k hodnote.

Aby mohol blok pracovať s prijatými dátami, uloží si ich najprv do statických dát, tak isto odosielané dáta sú najprv spracované a uložené do statických dát, až potom odoslané. Množstvo prijatých aj odoslaných dát je obmedzená na 255 bytov. Blok začína prijatím dát zo zariadenia a ich uložením do statických dát. Na to je najprv potrebné vytvoriť ANY Pointer Destination, ktorý funkcia AG_RECV potrebuje ako cieľ uloženia prijatých dát. ANY Pointer (obr. 3.3) v sebe obsahuje všetky informácie definujúce cieľ dát.



Obr. 3.3 - Štruktúra ANY Pointer

V tomto prípade kopírujeme jednotlivé byty, ich počet je určený vstupným parametrom Q_BYTES, DB je inštančné DB aktuálneho FB, takže je možné zadať nulu. Ako posledná časť je vložený pointer na vytvorené pole bytov RECEIVE_DATA, ktorý už v sebe obsahuje aj Area kód. Funkcia AG_RECV potrebuje okrem ANY Pointeru Destination ešte ConnectionID a adresu komunikačného modulu, ktoré získa z vstupných parametrov bloku. Ostatné parametre sú výstupy umožňujúce kontrolovať stav priebehu funkcie.

```
LAR1  P##Destination

L     W#16#1002                  // bytes
T     W [AR1,P#0.0]
L     #Q_BYTES                   // repetition
T     W [AR1,P#2.0]
L     0                          // instance DB
T     W [AR1,P#4.0]
L     P##RECEIVE_DATA            // pointer
T     D [AR1,P#6.0]

CALL  "AG_RECV"
 ID    :=#ID_CONN
 LADDR :=#LADDR
 RECV  :=#Destination
 NDR   :=#COMM_NDR
 ERROR :=#COMM_ERR_R
 STATUS:=#COMM_ST
 LEN   :=#COMM_LEN
```

Po vykonaní tohto kódu sa v statických dátach nachádzajú prijaté dáta, ktoré je potrebné postupne podľa definovaných parametrov nakopírovať na príslušné výstupy bloku. Ako prvé sa v cykle prekopírujú byty obsahujúce bitové statusy prijaté zo zariadenia. Počet cyklov je určený parametrom Q_ST_BYTES.

Kopírovanie prebieha pomocou nepriameho adresovania a začína z prvého prijatého bytu na prvý výstup OBJ_ST_00, pričom sa pri každom cykle obidve adresy zvýšia o jeden byte.

```
      LAR1  P##OBJ_ST_00                // Pointer na prvy vystup
      L     P##RECEIVE_DATA             // Pointer na prijate data
      T     #RecvData
      L     #Q_ST_BYTES                 // Pocet cyklov
CMD:  T     #LOOP                       // Ulozenie zostavajuceho poctu
cyklov
      L     DIB [#RecvData]             // Kopirovanie dat
      T     DIB [AR1,P#0.0]
      L     P#1.0
      L     #RecvData
      +D
      T     #RecvData                   // zvysenie adresy dat o 1 byte
      +AR1  P#1.0                       // zvysenie adresy vystupu o 1 byte
      L     #LOOP
      LOOP  CMD                         // kontrola poctu cyklov
```

Po dokončení všetkých cyklov sú teda všetky bitové signály prekopírované na príslušných výstupoch a v RecvData zostane pointer na nasledujúci byte v prijatých dátach. Tieto dáta už ale obsahujú číselné hodnoty, ktoré treba prekopírovať na výstupy VALUE. V tomto prípade je to už ale zložitejšie, pretože je potrebné rozlíšiť typ prijatého čísla a prekonvertovať na typ REAL. Na to môžme v prípade INT aj BYTE použiť príkaz DTR - DoubleIntegerToReal.

```
      LAR1  P##VALUE_00         // Pointer na prvy vystup
      L     P##DT_VALUE_00      // Pointer na vstup datoveho typu
      T     #Value
      L     41                  // Pocet cyklov (max. 41 hodnot)
con:  T     #LOOP               // Ulozenie zostavajuceho poctu cyklov
      L     DIB [#Value]        // nacitanie datoveho typu aktualnej hodnoty
      JL    DEF1
      JU    CMDN                // 0 = NONE
      JU    CMDI                // 1 = INT
      JU    CMDB                // 2 = BYTE
      JU    CMDR                // 3 = REAL
DEF1: BEU
CMDN: JU    snd                 // Pri DT_VALUE = 0 sa cyklus konci
CMDI: L     DIW [#RecvData]     // Nacitanie hodnoty z prijatych dat
      DTR                       // Prevod na real
      T     DID [AR1,P#0.0]     // Kopirovanie na aktualny vystup
      L     P#2.0               // Nacitanie posunu adresy
      JU    CMDC
CMDB: L     DIB [#RecvData]     // Nacitanie hodnoty z prijatych dat
      DTR                       // Prevod na real
      T     DID [AR1,P#0.0]     // Kopirovanie na aktualny vystup
      L     P#1.0               // Nacitanie posunu adresy
      JU    CMDC
CMDR: L     DID [#RecvData]     // Nacitanie hodnoty z prijatych dat
      T     DID [AR1,P#0.0]     // Kopirovanie na aktualny vystup
      L     P#4.0               // Nacitanie posunu adresy
```

V tomto prípade sa adresa na prijaté dáta posúva vždy o iný počet bytov, ktorý je daný dĺžkou definovaného dátového typu. Tento posun si teda vždy načítame do ACC1 tesne pred skokom na CMDC, kde túto hodnotu pripočítame k pointru RecvData.

```
CMDC: L     #RecvData
      +D
      T     #RecvData          // Pripocitanie nastaveneho posunu adresy
      NOP   0                  // k adrese prijatych dat
      +AR1  P#4.0              // Zvysenie adresy na dalsi vystup VALUE
      L     #Value
      L     P#1.0
      +D
      T     #Value             // Posun adresy na dalsi DT_VALUE
      L     #LOOP
      LOOP  con                // kontrola poctu cyklov
      JU    snd
```

Výstupy sú všetky typu REAL, takže tam je posun vždy 4 byty, vstupy DT_VALUE sú jednobytové. Po dokončení posledného cyklu sú teda na všetkých výstupoch rozkopírované prijaté dáta, ostatné výstupy sa nepoužívajú. Následne je potrebné pripraviť dáta pre odosielanie. Kopírovanie bitových signálov je rovnaké ako v prípade prijatých dát, akurát sa kopírujú v opačnom smere, zo vstupov do statických dát. Aj kopírovanie dát je podobné, rozdiely sú hlavne v opačnej konverzii z INT alebo BYTE na REAL. V tomto prípade totiž môže dôjsť k pretečeniu daného typu, aj keď by táto situácia nemala nastať, pre istotu to blok kontroluje. Kontrola spočíva v posunutí čísla o počet bitov, koľko má výsledný dátový typ, čím sa z neho odstránia všetky jeho relevantné bity. Pokiaľ sa po tejto operácii výsledok nerovná nule, znamená to, že pôvodné číslo bolo mimo rozsah požadovaného dátového typu.

```
PARI: L     DID [AR1,P#0.0]    // Nacitanie hodnoty zo vstupu
      RND                      // Zaokruhlenie = prevod REAL na INT
      T     DIW [#StData]      // Kopirovanie do dat na odoslanie
      SRD   16                 // Odstranenie 16 bitov (INT)
      L     0
      ==D                      // Porovnanie s 0
      JCN   ERR                // Kontrola pretecenia INT
      L     P#2.0              // Nacitanie posunu adresy
      JU    PARC

PARB: L     DID [AR1,P#0.0]    // Nacitanie hodnoty zo vstupu
      RND                      // Zaokruhlenie = prevod REAL na BYTE
      T     DIB [#StData]      // Kopirovanie do dat na odoslanie
      SRD   8                  // Odstranenie 8 bitov (BYTE)
      L     0
      ==D                      // Porovnanie s 0
      JCN   ERR                // Kontrola pretecenia BTYE
      L     P#1.0              // Nacitanie posunu adresy
      JU    PARC

PARR: L     DID [AR1,P#0.0]    // Nacitanie hodnoty zo vstupu
      T     DID [#StData]      // Kopirovanie do dat na odoslanie
      L     P#4.0              // Nacitanie posunu adresy
```

V prípade typu REAL sa samozrejme nič nekonvertuje a teda ani nekontroluje, zvyšok cyklu je obdobný ako v prípade prijatých dát. Nakoniec je opäť vytvorený ANY Pointer, tento krát pre zdrojové dáta, ktoré sú pomocou funkcie AG_SEND odoslané zariadeniu. Na konci bloku je ešte kód vyhodnocujúci stav a chyby pre zobrazeniu bloku vo vizualizácii.

3.2.2 Rozdelenie technológie na skupiny

Z hľadiska technologického riešenia je možné súčasne dopravovať materiál z násypky 8 do síl a vápenec prípadne zmes na skládku. Preto sú tieto dve časti v riadiacom systéme rozvrhnuté do dvoch skupín 10G10 a 10G20, ktorých beh je úplne nezávislý. Doprava čistého vápenca z násypky 3 priamo do sila 2 je možná iba osamotene, pretože používa dopravné cesty z obidvoch častí technológie. Tento prípad je riešený skupinou 10G30, ktorá môže byť spustená iba samotná.

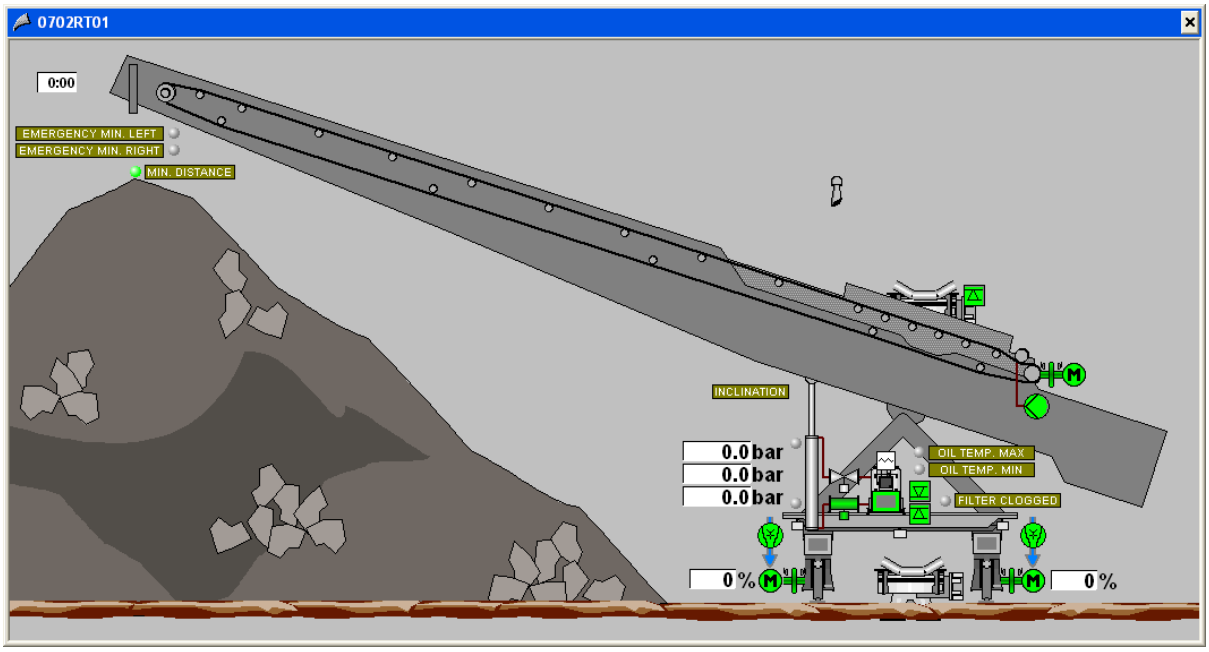
3.2.3 Skupina 10G10 - Vápenec + hlina

Táto hlavná skupina zaobstaráva transport vápenca a hliny z násypiek 3 a 5 na homogenizačnú skládku. Obsahuje podradenú skupinu 10G11 do ktorej patrí samotný transport a drvenie materiálu, skupiny odprášenia dopravných ciest 10G12-10G14 a skupinu odberu vzoriek 10G15. Skupina 10G11 obsahuje dve cesty, ktoré určujú zdroj a cieľ dopravy suroviny. Pri navolení cesty 10W11A je dopravovaná zmes vytvorená miešaním vápenca z násypky 3 s hlinou z násypky 5. Cesta 10W11B je určená na dopravu čistého vápenca z na skládku. Pri navolení tejto cesty sa nerozbieha časť dopravy od násypky 5 ani valcový drvič. Objekty tejto skupiny sú na obrazovke 0305_Limestone_Clay (príloha A).

Pri spúšťaní ako aj behu všetkých dopravníkov riadiaci systém zabezpečuje správne radové blokovanie dopravníkov. Pri detekcii preťaženia drvičov (nadlimitný prúd alebo príkon) sa drviče nezastavujú, znížia sa otáčky podávača do drviča, čím sa zníži množstvo materiálu pridávaného do drviča až do doby návratu hodnôt do normálneho stavu. Toto zníženie sme vyriešili redukciou setpointu na regulátore, takže v prípade potreby má operátor možnosť toto zníženie ovplyvniť (meniť percentuálne zníženie otáčok). Tak isto pri zastavení niektorého pásu sú drviče z blokácie vynechané, nie je totiž vhodné ich zastavovať pokiaľ je v nich materiál. Pri miešaní zmesi je kontrolovaný beh obidvoch drvičov, v prípade zastavenia (nespustenia) jedného z nich sa po 10 sekundách zastaví podávanie materiálu. Valcový drvič na hlinu má možnosť chodu v dvoch smeroch, tieto smery sa pri normálnom behu striedajú pri každom spustení, prípadne môže operátor navoliť požadovaný smer nasledujúceho spustenia. Množstvo hliny a vápenca v zmesi sa vypočítava a riadi podľa nastaveného žiadaného množstva zmesi a percentuálneho obsahu hliny, pričom aktuálne žiadané množstvo hliny je vypočítavané zo skutočnej váhy zmesi, čím sa zabezpečí správne dávkovanie hneď po spustení dopravy. Z pásu dopravujúceho zmes sa odoberajú vzorky, ktoré sa odvážajú do laboratória na analýzu zloženia zmesi. Vzorkovanie zabezpečuje skupina 10G15.

Po rozdrvení a správnom zmiešaní zmesi sa dopravníkmi transportuje na homogenizačnú skládku, ktorá sa skladá z troch háld. Na dve haldy (A a B) sa zakladá zmes, tretia (C) je určená pre čistý vápenec. Vizualizačná obrazovka predhomogenizačnej skládky 07_Preblending je v prílohe B.

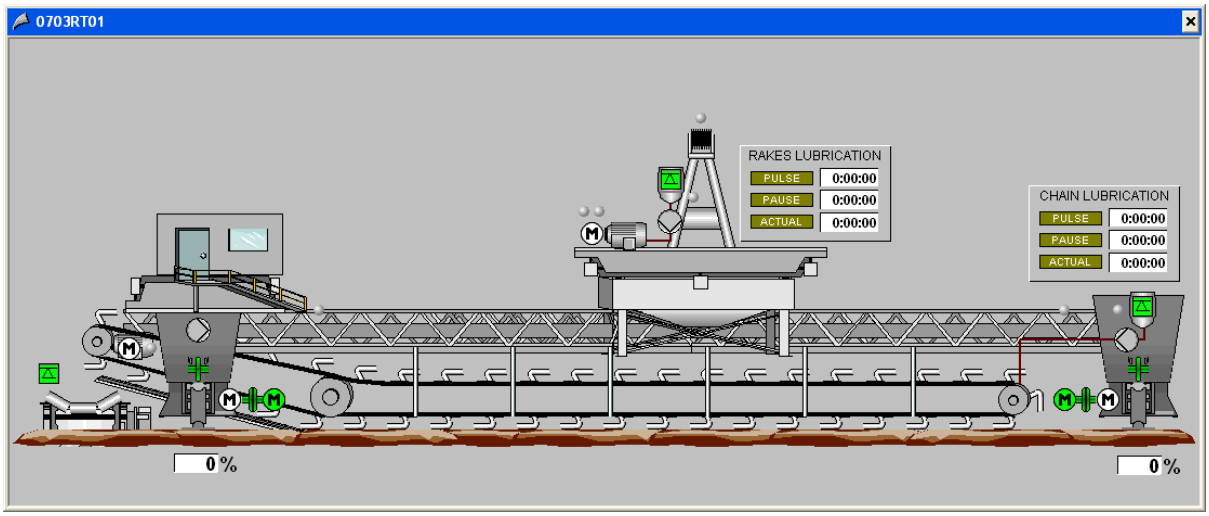
Samotný zakladač je riadený vlastným PLC, a komunikuje s nadradeným systémom pomocou vytvoreného bloku U_EXT_DEVICE. Pomocou tejto komunikácie nadradený systém posiela povely a číselné parametre pre zakladač a dostáva od neho informácie o stave jednotlivých zariadení. Tieto objekty sú vďaka tomu zobrazené vo vizualizácii, aj keď ich nie je možné riadiť. Pre lepšie zobrazenie stavu týchto objektov na vizualizácii sme vytvorili detailné vyskakovacie okno (obr. 3.4).



Obr. 3.4 - Detail zakladača

3.2.4 Skupina 10G20 - Transport suroviny

Skupina 10G20 zaobstaráva transport zmesi z homogenizačnej skládky (patrí do nej aj škrabák) a surovín z násypky 8 do síl pred surovinovým mlynom. Skupina obsahuje podskupinu transportu 10G21 a tri skupiny odprášenia 10G22 - 10G24. Celkovo obsahuje 4 možnosti dopravy, pričom tri z nich sú z násypky, ale vždy do iného sila. Medzi týmito možnosťami sa prepína pomocou ciest 10W21A - 10W21D. Pre zamedzenie chýb operátora a nechcené nasypanie materiálu do nesprávneho sila tieto cesty definujú jednoznačne zdroj aj cieľ. Jedným zo zdrojov materiálu v tejto skupine je škrabák, ktorý je rovnako ako zakladač riadený vlastným PLC. Jeho detailná vizualizácia je na obr. 3.5.



Obr. 3.5 - Detail škrabáku

Najväčším problémom tejto skupiny bolo vyriešenie zmeny cesty za behu skupiny, pretože štandardne sa cesty prepínajú buď za behu (všetkých) pohonov, alebo pri vypnutých pohonoch. V tomto prípade bolo požadované prepínanie zložitejšie, vypínať sa majú iba pohony, ktoré nebudú aktívne v novozvolenej ceste. Zníži sa tým opotrebovanie týchto pohonov vyplývajúce z častejšieho zastavovania a spúšťania. Navyše je vždy treba dodržať pre každý dopravník rôzny čas dobehu pre vyčistenie. Pri prevolení cesty sa zmene prispôsobuje aj odprášenie, zapínajú alebo vypínajú sa príslušné skupiny odprášenia podľa toho, ktoré pásy sú v danej ceste použité, navyše je ale nutné riadiť otváranie klapiek na vstupoch odprášenia tak, aby bola pre dané odprášenie otvorená vždy iba jedna. Pri otvorení obidvoch by odsávací výkon nestačil. Vizualizácia tejto časti je na obrazovke 08_Raw_transport (príloha C).

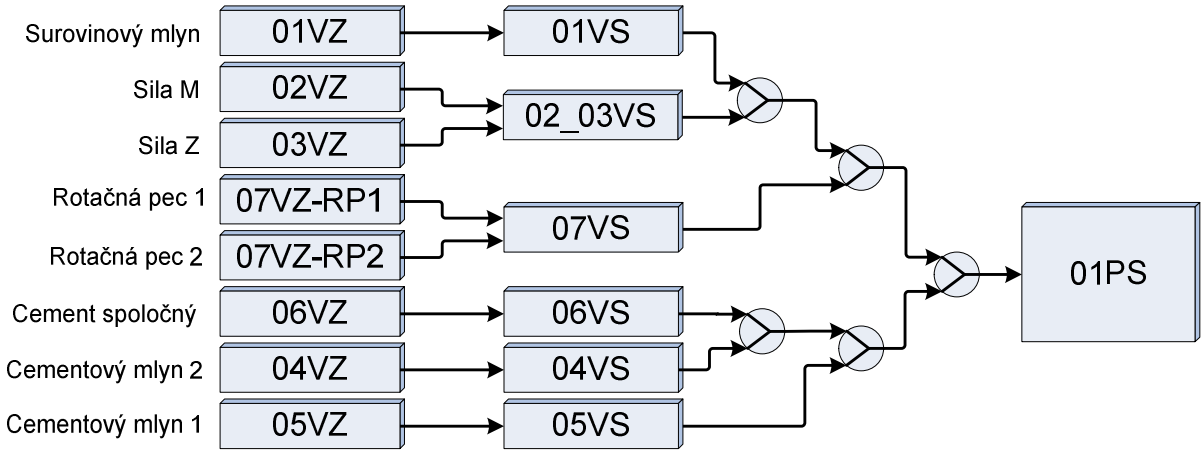
3.2.5 Skupina 10G30 - Transport čistého vápenca do sila

Transport čistého vápenca z násypky 3 do sila 2 je obstarávaný skupinou 10G30, ktorá obsahuje súčasne niektoré podskupiny skupín 10G10 a 10G20. Preto je nutné zabezpečiť jej samostatný beh. Aby bol prechod z chodu skupín 10G10 a 10G20 na skupinu 10G30 čo najrýchlejší a najekonomickejší, navrhli sme systém tak, že nie je nutné vypínať celé hlavné skupiny, pretože podskupiny odprášenia sa vypínajú 10 minút a potom by sa hneď zase zapínali, čo je úplne zbytočné. Na spustenie skupiny 10G30 teda stačí vypnúť skupiny transportu (10G11 a 10G21), spustenie potrebných nebežiacich skupín odprášenia si zabezpečí hlavná skupina automaticky. Pri prevoľovaní skupín sa tiež zmení cesta odpraškov zo skupiny odprášenia 10G13, podľa toho, ktorý dopravný pás je spustený. Samotný transport spúšťa podskupina 10G31. Spustenú skupinu 10G30 vidieť celú na prehľadovej obrazovke Crushing (príloha D).

4. Projekt Radotín - Automatické laboratórium

4.1 Pôvodný stav

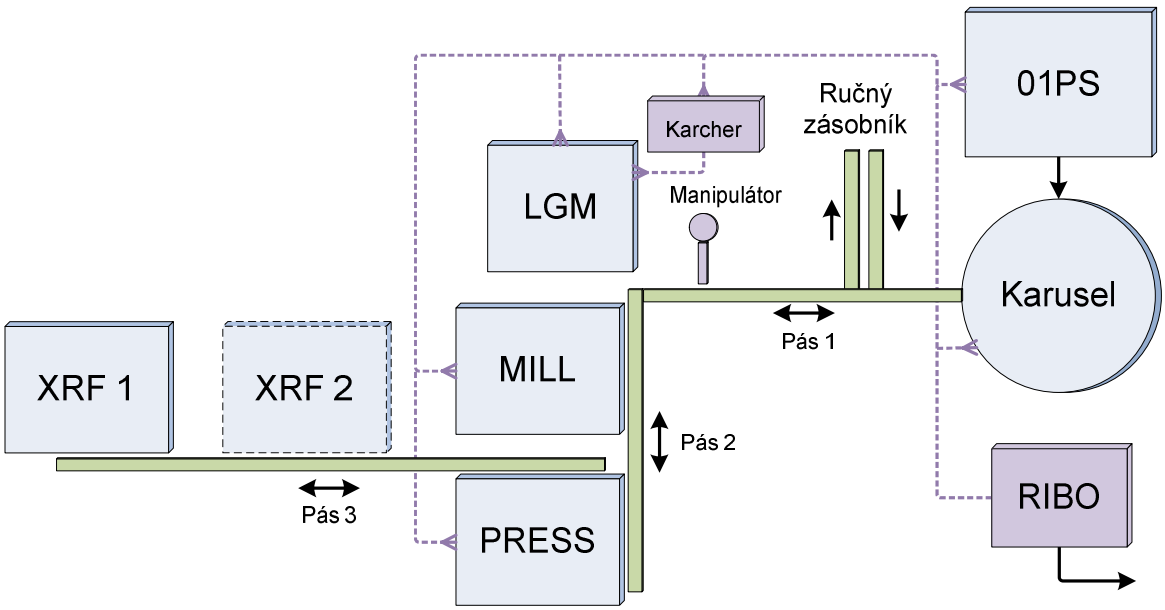
Laboratórium v cementárni slúži na analyzovanie zloženia materiálu v priebehu celého procesu výroby cementu. Každé odberné miesto pozostáva zo vzorkovačky 0xVZ, miešača a vysielacej stanice 0xVS. Vzorkovačka odoberá vzorky materiálu postupne po menších dávkach do miešača. Tým sa získa vzorka obsahujúca materiál z dlhšieho časového obdobia a zmiernia sa možné extrémy v analyzovaných vzorkách. Po zozbieraní dostatočného počtu dávok a dosiahnutí požadovaného času je možné naplniť patrónu a odoslať ju pomocou vysielacej stanice do laboratória. Niektoré vzorkovačky sú umiestnené fyzicky blízko seba, takže zdieľajú spoločnú vysielaciu stanicu. Vzorky sú z vysielacích staníc dopravované potrubnou poštou v uzatvorených patrónach do prijímacej stanice. Schéma tejto časti je na obr. 4.1. Každá vysielacia stanica má vlastnú patrónu, a v potrubnej pošte a prijímacej stanici sa môže vždy nachádzať iba jedna. To znamená že po odoslaní patróny ostatné stanice čakajú, kým sa táto patróna nevyprázdni v prijímacej stanici a nedorazí naspäť, až potom je možné odoslať pripravenú patrónu z inej vysielacej stanice.



Obr. 4.1 - Schéma vzorkovačiek a potrubnej pošty

V prijímacej stanici sa patróna otvorí, vysype do vykladača, ktorý nasype požadované množstvo vzorky do priradených kalíškov a zberných nádob umiestnených na karuseli. Vzorky v nádobách sú dlhodobé, odoberajú a analyzujú sa ručne. Kalíšky so vzorkami sú analyzované automaticky podľa nastavení programu. Analýza materiálu prebieha buď v lasergranulometri (LGM), alebo v röntgene (XRF1 alebo XRF2).

Bloková schéma usporiadania laboratória je na obr. 4.2. Vzorky určené pre LGM sú dopravované po páse 1 pred manipulátor, ktorý uchytí kalíšok a vysype ho do analyzátoru. Prázdny kalíšok potom putuje naspäť na pôvodné miesto v karuseli, kde sa následne vyčistí. Vzorka určená pre analýzu v XRF je dopravená po páse 1 a 2 do mlynu, kde sa zomelie na požadovanú jemnosť. Kalíšok so zomletým materiálom ďalej putuje po páse 2 do lisu, kde sa materiál vlisuje do kovového krúžku. Vyprázdnený kalíšok putuje naspäť po páse 1 a 2 do karuselu, kde sa vyčistí. Krúžok so vzorkou je po páse 3 dopravený do röntgenu XRF 1, kde sa spustí analýza. Po jej dokončení sa krúžok vráti späť do lisu a tam sa vyčistí. XRF 2 je nový, je využitý až v novom systéme.



Obr. 4.2 - Bloková schéma usporiadania laboratória

Tak ako z karuselu, je možné analyzovať aj vzorky z ručného podávača. Ten sa využíva napríklad na analýzu materiálu z lomov, reanalýzu vzoriek z technológie alebo analýzu materiálu z nádob na dlhodobé vzorky. Cesta kalíškov je takmer rovnaká ako z karuselu, s tým rozdielom, že kalíšky sa po vyprázdnení nevracajú na pôvodné miesto, ale na druhý pás ručného zásobníku a nečistia sa automaticky. Ďalším dôležitým zariadením je RIBO, ktoré zabezpečuje odprášenie laboratória. Spúšťa sa vždy keď je v činnosti niektoré zo zariadení, ktoré je na RIBO napojené. S odprašovaním súvisí aj vysávač Kärcher, ktorý slúži na vytvorenie podtlaku pri meraní LGM a tiež na čistenie jeho meracej komory po analýze. Materiál nazbieraný v zásobníku Kärcheru odsáva RIBO vždy po šiestich meraniach LGM.

Pôvodný riadiaci systém bol tvorený obyčajným PC s OS Linux a PLC Siemens rady S7-400. Na PC bežal program AMLAS, ktorý zabezpečoval hlavné funkcie riadenia, časť vizualizácie a rozhranie pre nastavenie systému a zadávanie vzoriek. Historické údaje o vzorkách sa uchovávali tiež na tom istom PC. Program v PLC obsahoval len základné sekvencie pre riadenie vzorkovačiek, miešačov a pohybu kalíškov, ich spúšťanie a riadenie mal na starosti AMLAS. Vizualizácia laboratória bola rozdelená na týchto dvoch systémoch, čo tiež neprispievalo k jednoduchej obsluhe.

4.2 Rozbor problémov

Základným problémom celého laboratória je použitý riadiaci systém. Nejedná sa totiž o žiadnu štandardnú riadiacu platformu ale o špecifický program vyvinutý na riadenie takýchto laboratórií. Systém ako taký nebol zlý, niekoľko rokov fungoval, problém bol s jeho rozšírením o nové funkcie alebo zariadenia. Práve z dôvodu častých porúch röntgenu a dĺžky jeho servisu bolo potrebné kúpiť druhý, ktorý by sa dal použiť ako náhrada. Rovnaký röntgen ako pôvodný už ale nebolo možné kúpiť, takže by bolo nutné pre nový upraviť aj riadiaci program AMLAS. Jeho výrobca už úpravy takto starého programu nerobí a jediná možnosť by bola kúpiť nový systém pre celé laboratórium. Ďalšou nevýhodou je potreba samostatných monitorov pre operátorov, jeden pre vizualizáciu AMLASu, a druhý pre vizualizáciu programu PLC, nakoľko tento systém je postavený na starom systéme CEMAT 4 a ten nie je možné zaintegrovať k ostatným častiam vizualizácie v PCS 7. Operátori tak mali na dvoch monitoroch stále AMLAS a vizualizáciu laboratória a nemohli monitory využiť na zobrazenie iných, aktuálne potrebných častí vizualizácie, tak ako sú zvyknutí z WinCC.

Z pohľadu na tieto problémy sa rozhodlo, že najlepším riešením bude kompletné prerobenie riadiaceho systému na platforme PCS 7 a jeho začlenenie do centralizovaného riadiaceho systému celej cementárne. Tým sa vyriešia všetky predchádzajúce nedostatky, nový systém bude štandardný, ľahko rozšíriteľný a vizualizácia bude súčasťou štandardnej vizualizácie celého projektu vo WinCC. Pre jednoduchšiu obsluhu chceli operátori do vizualizácie začleniť možnosť zistiť kde sa nachádzajú jednotlivé vzorky, prípadne aké vzorky sa aktuálne spracovávajú v jednotlivých zariadeniach. To bolo v starej vizualizácii prakticky nemožné, pretože program v PLC nemal informáciu s akou vzorkou pracuje. V novom systéme prevezme PLC podstatne viac práce, hlavne kompletný manažment práce so vzorkami, takže vyhovieť tejto požiadavke bude možné.

Z problémov funkčnosti samotného laboratória je najzávažnejšie zdieľanie dopravných pásov pre viaceré činnosti. Starý riadiaci systém mal toto zdieľanie vyriešené jednoducho, avšak nevhodne - keď poslal kalíšok do LGM, ďalší kalíšok do mlynu čakal a odoslal sa až keď sa z LGM vrátil prázdny kalíšok naspäť do karuselu. Takéto riešenie celý systém veľmi spomaľovalo a spôsobovalo vynechanie niektorých pravidelných vzoriek. Zdieľanie pásu 2 bolo vyriešené lepšie (rýchlejšie), ale občas sa vyskytol konflikt a kalíšky sa na páse stretli. V takom prípade sa kalíšky mohli vymeniť a do lisu sa mohol dostať nepomletý materiál, následkom čoho sa potom z nalisovaného krúžku materiál vysypal.

Ďalšou zdieľanou časťou je karusel, ktorý sa natáča podľa požiadaviek na pozíciu dávkovania, vykladania kalíšku, a čistenia. Starý systém fungoval na princípe prvý berie, čo zase spomaľovalo dávkovanie a celú potrubnú poštu. Podľa našich pozorovaní by pomohlo, keby malo dávkovanie vyššiu prioritu ako ostatné činnosti, takže by sa vykladač vyprázdnil rýchlejšie a prijímacia stanica by bola pripravená na príjem ďalšej patróny skôr.

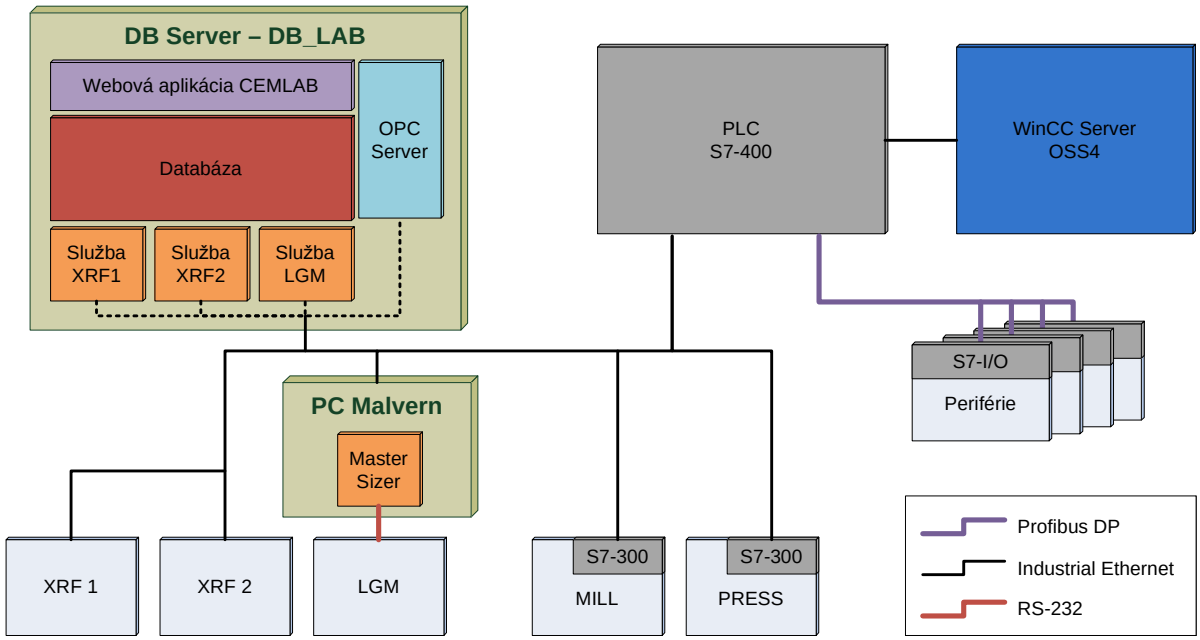
Občasné zastavenie činnosti spôsoboval aj nedostatok materiálu v kalíšku pre analýzu v XRF. Systém pôvodne obsahoval iba snímač dostatku materiálu pre analýzu v LGM, toto množstvo je však nedostatočné pre mlyn, ktorý potrebuje približne dvojnásobok materiálu. Na túto kontrolu má mlyn vlastný snímač dostatku materiálu, ktorý slúži hlavne na jeho ochranu pred poškodením pri práci s malým množstvom materiálu. Problém bol hlavne v tom, že pri detekcii nedostatku bol už kalíšok v mlyne a jediná možnosť ako systém znova spriechodniť bola ručne prepnúť mlyn do manuálneho režimu a kalíšok vybrať. Preto sme do karuselu pridali ďalší snímač nastavený na množstvo materiálu do mlynu, vďaka čomu kalíšok s nedostatkom materiálu neopustí karusel a riadiaci program ho sám vyčistí bez akéhokoľvek zásahu operátora.

Problém bol aj s odsávaním RIBO, ktoré je tiež zdieľané, ale môže súčasne odsávať viac zariadení. V starom systéme to bolo využité len čiastočne, v tejto oblasti bol tiež priestor na urýchlenie systému. Hlavný problém bol však s jeho vyprázdňovaním, ktoré je potrebné vykonať po určitom počte odsatí a počas tejto činnosti sa nesmie odsávať žiadne zariadenie (odsávanie by bolo neúčinné). Samozrejme už spustené odsávanie zariadení sa musí dokončiť a až potom sa môže RIBO vysypať. S odsávaním je spojený ďalší problém starého systému, ktorý niekedy nasypal vzorku do LGM ešte počas jeho čistenia, a tá bola celá odsatá a teda stratená.

4.3 Realizácia

4.3.1 Nová koncepcia systému

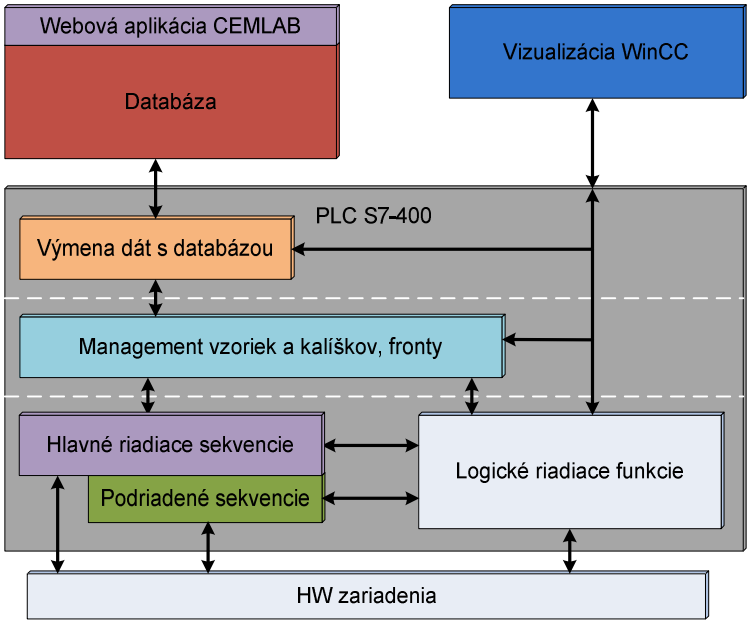
Srdcom celého systému riadenia laboratória je PLC Siemens rady S7-400 spolu so serverom DB_LAB, ktorých komunikáciu zabezpečuje OPC Server (Object linking and embedding for Process Control). PLC je samozrejme súčasťou multiprojektu PCS 7, takže komunikuje s WinCC serverom, ktorý zabezpečuje vizualizáciu procesu. Bloková schéma komponentov riadiaceho systému je na obr. 4.3.



Obr. 4.3- Bloková schéma komponentov riadiaceho systému a ich komunikácie

Server DB_LAB obsahuje databázu, ktorá má spolu s webovou aplikáciou CEMLAB na starosti históriu vzoriek a uchovávanie nastavení a parametrov pre zariadenia v laboratóriu. Ďalej na serveri bežia služby pre komunikáciu a riadenie XRF1 a XRF2, LGM obsluhuje program MasterSizer dodaný výrobcom, s ktorým tiež komunikuje služba na serveri. Tieto služby zabezpečujú samotné povelovanie zariadení, parametrizovanie a predávanie výsledkov meraní do databázy. K hlavnému PLC sú cez Profibus DP pripojené distribuované I/O moduly pre celé laboratórium a všetky vzorkovačky. Mlyn a lis sú riadené menšími PLC Siemens rady S7-300 pripojené k hlavnému PLC cez Industrial Ethernet. Tieto PLC sú dodané a naprogramované priamo výrobcom pôvodného systému, takže ani k nim nemáme žiadnu dokumentáciu a program. Ten sa síce dá stiahnuť z PLC, avšak je bez symboliky a tým pádom veľmi neprehľadný a náročný na pochopenie a úpravy. Museli sme sa teda prispôsobiť a snažiť sa použiť zariadenia tak ako sú.

Program v hlavnom PLC je rozdelený na niekoľko vrstiev (obr. 4.4), z ktorých každá má svoje špecifické úlohy. Najvyššia vrstva zabezpečuje komunikáciu s databázou, získavanie ID pre nové vzorky, načítanie dát o existujúcich vzorkách pri ich spracovaní a aktualizáciu ich stavu. Údaje získané z databázy využíva nižšia vrstva, ktorá sa stará o celý manažment vzoriek, ich odosielanie v správnom čase, rozdelenie vzorky na kalíšky a ich správne zoradenie podľa priorít. Táto vrstva potom dáva povely najnižšej vrstve, ktoré už priamo ovplyvňujú HW zariadenia.

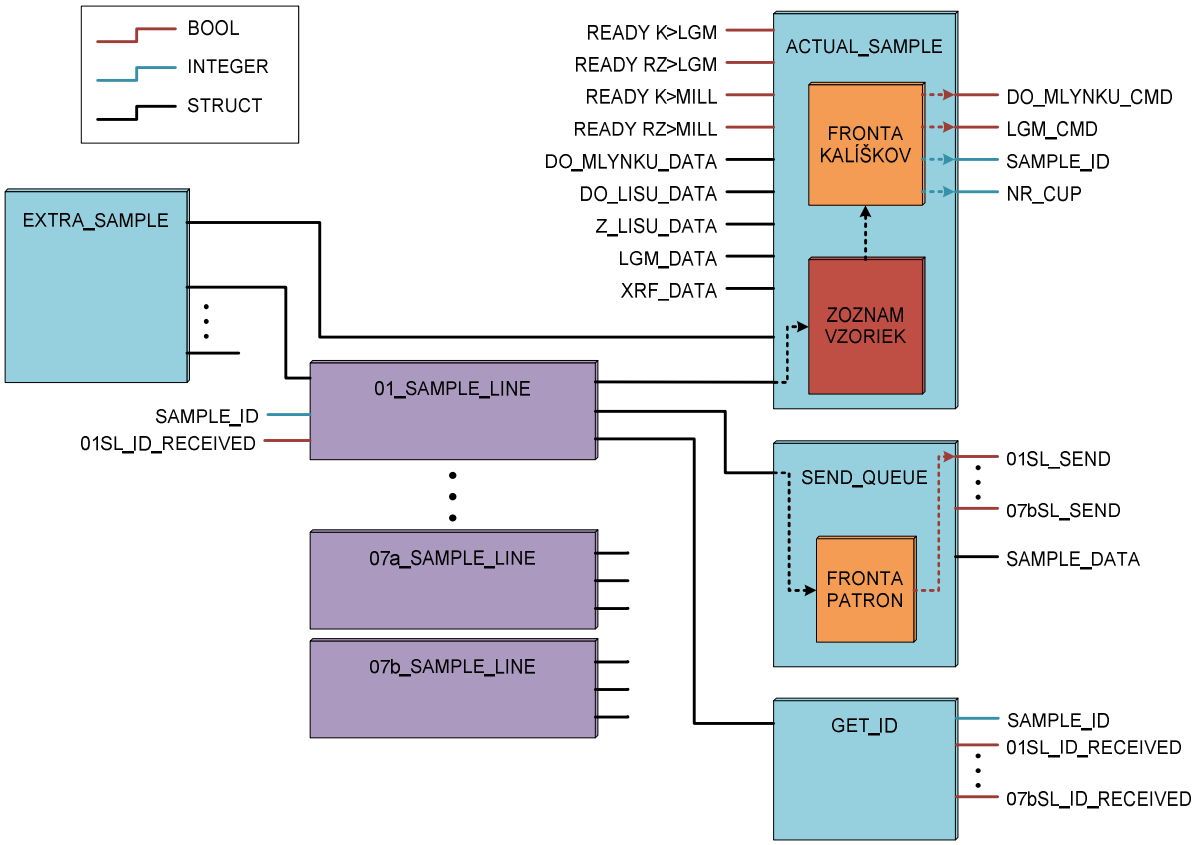


Obr. 4.4 - Štruktúra riadiaceho systému v PLC

Najnižšia vrstva v sebe obsahuje hlavné a podriadené riadiace sekvencie a logické riadiace funkcie. Hlavné sekvencie sa starajú o riadenie základných činností systému - vzorkovanie materiálu, príprava, odoslanie a prijatie patróny, jej rozdelenie do kalíškov a následnú dopravu do zariadení na analýzu. Tieto sekvencie ovládajú priamo niektoré ventily a pohony, ale tiež využívajú služieb podriadených sekvencií a časť logických riadiacich funkcií. Podriadené funkcie slúžia na riadenie menších celkov, ktoré sa využívajú vo viacerých hlavných sekvenciách, alebo riadia samostatné funkčné celky.

Všetky vrstvy majú vytvorenú vizualizáciu, takže je možné v akomkoľvek momente zistiť, čo sa s každou vzorkou deje a kde sa nachádza. Vizualizácia WinCC slúži hlavne na monitoring, prípadne riešenie poruchových stavov. Na užívateľský vstup je určená webová aplikácia zabezpečujúca všetky nastavenia vzorkovania, parametrov jednotlivých materiálov, vytváranie mimoriadnych vzoriek a tiež zobrazovanie výsledkov analýz a reportov.

Manažment vzoriek a kalíškov sa skladá z niekoľkých funkčných blokov (FB) vytvorených špeciálne pre tento účel. Ich blokový diagram je na obr. 4.5.



Obr. 4.5 - Blokový diagram hlavných riadiacich blokov

SAMPLE_LINE - Každá vzorkovačka má vlastnú instanciu tohto bloku, ktorý riadi vykonávanie pravidelných odberov materiálu ako aj mimoriadne vzorky. Pravidelné odbery sa vykonávajú na základe časov nastavených v databáze, odkiaľ si ich tento blok prevezme spolu s materiálovým kódom a údajmi kam sú vzorky určené (čísla kalíškov a nádoby).

SEND_QUEUE - Tento blok sa stará o správne radenie a odosielanie naplnených patron z jednotlivých vzorkovačiek do laboratória. Z princípu je to fronta s radením podľa priorít.

GET_ID - Zabezpečuje získanie ID pre každú novú vzorku, či už ide o pravidelnú vzorku zo vzorkovačiek, alebo mimoriadnu vzorku vyžiadanú operátorom. ID vzoriek si vytvára a spravuje databáza, bez priradeného jedinečného ID by sa vzorka nemohla správne identifikovať a mohlo by dochádzať k chybám pri jej spracovaní.

EXTRA_SAMPLE - Zabezpečuje pridanie mimoriadnych vzoriek zadaných užívateľom cez webovú aplikáciu CEMLAB. Mimoriadnu vzorku je možné vytvoriť z každej vzorkovačky, ako aj z ručného zakladača priamo v laboratóriu. Preto je tento blok napojený na všetky bloky SAMPLE_LINE, aj do bloku ACTUAL_SAMPLE

ACTUAL_SAMPLE - Tento blok riadi celé spracovanie vzoriek, od ich vytvorenia, cez rozdelenie do kalíškov až po ich odoslanie na analýzu. Z hľadiska hierarchie je tento blok nadradený ostatným, akékoľvek operácie s existujúcimi vzorkami riadi práve on. Preto obsahuje zoznam všetkých vzoriek prítomných v systéme. Z tohto zoznamu sa vzorka vymaže až po dokončení jej analýzy a uložení výsledkov do databázy, prípadne po ručnom zmazaní. Práve pri ručnom zmazaní vzorky je tento blok dôležitý, pretože z informácií ktoré uchováva o každej vzorke vie určiť kde sa aktuálne nachádza a zabezpečiť nie len jej zmazanie zo systému, ale aj ukončenie jej spracovania - napríklad vrátenie a vyčistenie kalíšku bez analýzy. Ďalej tento blok obsahuje frontu kalíškov, ktorá radí kalíšky podľa nastavených priorít a podmienok v laboratóriu a riadi ich odosielanie na analýzu. Práve od tohto bloku dostávajú povely hlavné sekvencie na riadenie pohybu kalíškov v laboratóriu.

4.3.2 Riadenie laboratória

Samotné laboratórium je rozdelené na štyri hlavné časti - prijímacia stanica, odsávanie (RIBO), karusel a preprava kalíškov. Riadenie všetkých týchto častí je prevažne sekvenčné. Pri návrhu sekvencií som sa snažil vychádzať z pôvodného rozdelenia, na ktoré boli operátori zvyknutí. Zmenu prístupu si vyžadovala hlavne časť prepravy kalíškov, ktorú bolo potrebné zrýchliť, čo znamenalo kompletne novú koncepciu týchto sekvencií.

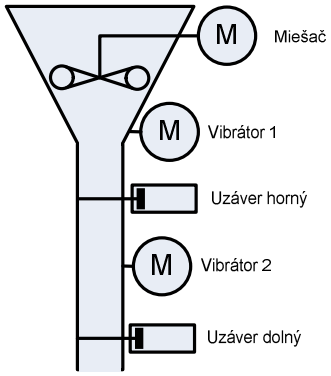
4.3.3 Prijímacia stanica

Riadenie prijímacej stanice pozostáva z troch sekvencií:

PS - Po prijatí patróny z potrubnej pošty (dosadnutí na snímač) pomocou sústavy piestov riadených ventilmi postupne uchopí patronu a jej uzáver, ktorý uvolní a zdvyhne. Tým je patrona s materiálom otvorená a materiál sa vysype do zásobníku vykladača. Týmto sa spustí sekvencia VYKLADAC, ktorá vysunie vykladač nad karusel. Tým sa vytvorí priestor pre vyčistenie patróny, jej opätovné uzatvorenie a odoslanie späť do vysielacej stanice. Po odoslaní sekvencia čaká na dokončenie sekvencie VYKLADAC a následne sa vráti do čakacieho stavu, kde je pripravená na prijatie ďalšej patróny.

VYKLADAC - Po prijatí povelu od sekvencie PS vysunie vykladač nad karusel, kde po jeho natočení na požadovanú pozíciu (kalíšok alebo nádobu) nasype časť vzorky (pomocou sekvencie DAVKA). Toto sa opakuje na všetkých pozíciách, ktoré sú zadané v databáze (napr. kalíšok LGM, kalíšok XRF, nádoba). Pretože niektoré vzorky sa analyzujú iba v XRF a z iných sa nevytvára dlhodobá vzorka v nádobe, je počet dávkovaní rôzny podľa zdroja vzorky. Po dokončení dávkovania na všetkých pozíciách sa vykladač natočí naspäť do prijímacej stanice kde sa odsaje zvyšok vzorky a zásobník sa dôklade vyčistí.

DAVKA - Sa stará o presné dávkovanie požadovaného množstva materiálu do kalíšku alebo nádoby. Čo najpresnejšie dávkovanie je dôležité hlavne pre dlhodobé vzorky v nádobách, aby sa zabezpečilo konštantné množstvo jednotlivých čiastočných vzoriek. Princíp dávkovania je znázornený na Obr. 4.6. Po vysunutí vykladača dostane od sekvencie VYKLADAC povel na dávkovanie a požadovaný počet dávok. Sekvencia spustí vykladač a po jeho dosadnutí na kalíšok alebo hrdlo nádoby sa otvorí horný uzáver, čím sa časť materiálu dostane do dolného priestoru medzi uzávermi. Potom sa na 4 sekundy spustí miešač a následne na ďalšie 4 sekundy vibrátor 1, aby sa materiál odlepil zo stien vykladača. Potom sa zatvorí horný uzáver a zastaví sa miešanie a klepanie.

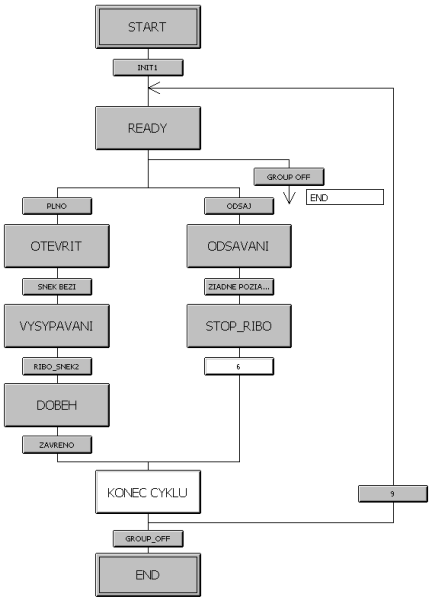


Obr. 4.6 - Princíp dávkovania materiálu

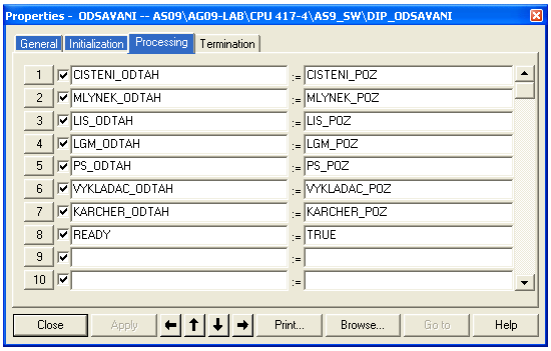
Sekvencia pokračuje otvorením dolného uzáveru, vďaka čomu sa do kalíšku vysype jedna dávka materiálu. Aby v zásobníku nezostal nalepený žiadny materiál, spustí sa vibrátor 2. Po uplynutí času 4 sekundy sa zavrie dolný uzáver, tým je dokončená jedna dávka. Pokiaľ má byť do kalíšku alebo nádoby nasypaných viac dávok, celá táto časť sa opakuje až do splnenia požadovaného počtu. Následne sa vykladač zodvihne do počiatočnej polohy a sekvencia končí.

4.3.4 Odsávanie RIBO

Pretože pri práci s navzorkovaným materiálom dochádza k rozvíreniu prachu, je nutné tento prach odsávať. Na to je v laboratóriu spoločný systém odsávania RIBO, ktorý je dosť silný na odsávanie všetkých zariadení. Odsatý materiál sa hromadí v sile a po určitom počte cyklov odsávania sa musí vysypať. Počas tejto doby je odsávanie neúčinné, takže nesmie pracovať žiadne zariadenie, ktoré by to potrebovalo. To bol jeden z problémov starého systému, ktorý niekedy začal vysypávanie počas práce mlynu alebo lisu, v ktorých potom nahromadený prach spôsoboval problémy. Aby som tomu zabránil, navrhol som sekvenciu pre odsávanie (obr. 4.7) tak, aby odsávanie a vysypávanie boli dve samostatné vetvy sekvencie, takže je nemožné aby sa spustili obidve súčasne.



Obr. 4.7 - Sekvencia riadenia odsávania



Obr. 4.8 - Akcie v kroku ODSAVANI

Po spustení sekvencia čaká v kroku READY, kým nedostane povel na odsávanie. Po prijatí povelu prejde do kroku ODSAVANI, kde spustí odsávanie a regeneráciu. Počas odsávania jedného zariadenia môže o odsatie požiadať aj akékoľvek iné zariadenie, takže v tomto kroku zostáva až do zrušenia všetkých povelov. Pri odsávaní LGM a Kärcheru je ešte nutné otvoriť príslušné ventily, na čo som využil akcie *Processing*, v ktorých na výstupy priraďujem hodnotu vstupných povelov jednotlivých zariadení (obr. 4.8). Tým dosiahnem otvorenie ventilov od týchto výstupov aj keď príde povel počas vykonávania kroku. Keď sa dosiahne požadovaný počet cyklov odsávania, zruší sa výstup Ready, takže žiadna sekvencia už nepošle povel, ale už bežiace odsávanie sa dokončí. Potom sa sekvencia vráti do stavu READY, odkiaľ pokračuje do vetvy vysypávania. Po dokončení je opäť možné odsávanie.

4.3.5 Riadenie presunu kalíškov

Jedným z hlavných obmedzení starého systému bola rýchlosť prepravy kalíškov so vzorkami z karuselu do jednotlivých zariadení. Aby sa táto preprava čo najviac urýchlila, rozdelil som riadenie na menšie celky, ktoré sa starajú o jednotlivé časti a kalíšky si medzi sebou predávajú:

- DO_MLYNKU - doprava kalíšku z karuselu alebo ručného podávača do mlyna
  - načítanie a predanie parametrov pre mlyn
  - odovzdanie kalíšku po ukončení mletia sekvencii DO_LISU
  - vyčistenie kalíšku (v prípade nedostatku materiálu v kalíšku)

- DO_LISU - doprava kalíšku z mlynu do lisu
  - načítanie a predanie parametrov pre lis
  - zabezpečenie rozdelenia kalíšku a vzorky a ich predania daným sekvenciám

- Z_LISU - doprava prázdneho kalíšku z lisu do karuselu alebo ručného podávača
  - vyčistenie kalíšku (v prípade uloženia do karuselu)

- XRF - doprava krúžku so vzorkou z lisu do zvoleného röntgenu
  - dopravu krúžku z röntgenu do lisu po analýze

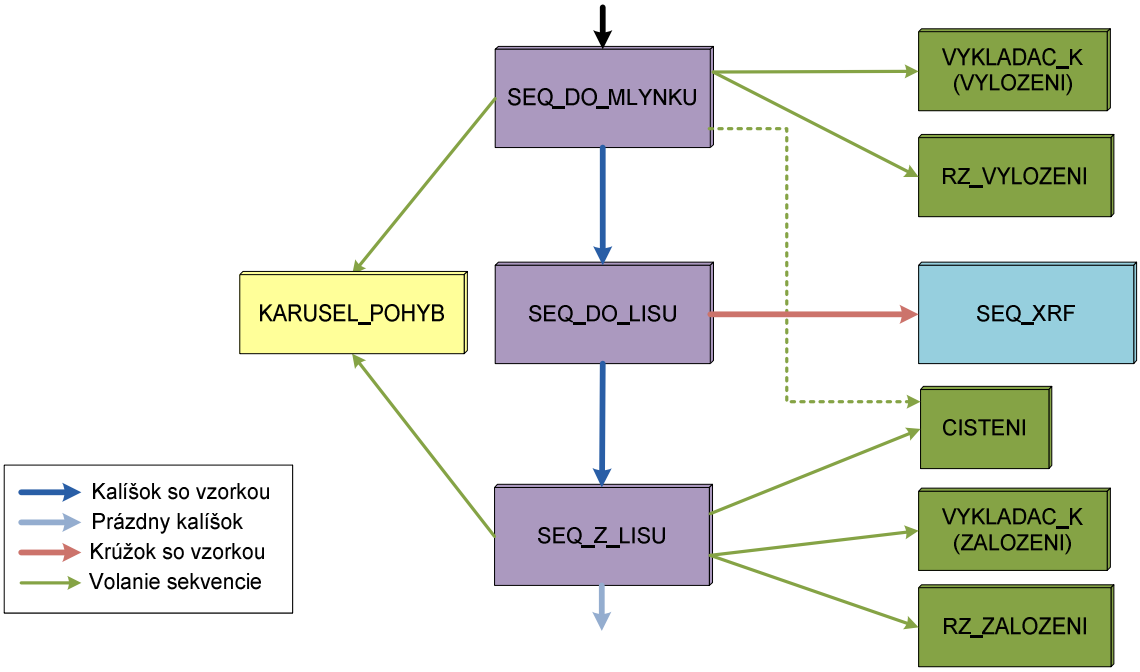
- LGM - doprava kalíšku karuselu alebo ručného podávača k manipulátoru
  - predanie kalíšku sekvencii MANIPULATOR_LGM
  - dopravu prázdneho kalíšku späť do karuselu alebo ručného podávača
  - vyčistenie kalíšku (v prípade uloženia do karuselu)

- MANIPULATOR_LGM - premiestnenie kalíšku nad násypku LGM
  - vysypanie kalíšku do násypky
  - vrátenie kalíšku na pás

Aby neboli tieto sekvencie príliš zložité a vzhľadom na opakujúce sa činnosti som ešte vytvoril niekoľko podriadených sekvencií, ktoré vykonávajú čiastočné úlohy a sú spúšťané od hlavných sekvencií - RZ_VYLOZENI, RZ_ZALOZENI, VYKLADAC_K a CISTENI.

Postup sekvencií pre cestu do XRF je na obr. 4.9. Sekvencia DO_MLYNKU po prijatí príkazu od fronty prevezme kalíšok a rozdelí sa na dve synchrónne vetvy. Jedna je zodpovedná za samotný pohyb kalíšku, druhá zabezpečuje načítanie parametrov pre mlyn, ktoré sú uložené pre každý typ materiálu v databáze.

Ďalej sa prvá vetva delí podľa zdroja kalíšku (karusel alebo ručný podávač). V prípade ručného podávača spustí sekvenciu RZ_VYLOZENI, ktorá vyloží jeden kalíšok zo zásobníku na pás 1. V prípade karuselu ho musí najprv pomocou sekvencie KARUSEL_POHYB natočiť na správnu pozíciu a skontrolovať prítomnosť kalíšku a dostatok materiálu pre mlyn. V prípade nedostatku materiálu sa vzorka vyhodnotí ako chybná a kalíšok sa pošle na vyčistenie. Ak je materiálu dostatok, spustením sekvencie VYKLADAC_K sa zaistí jeho vyloženie z karuselu na pás.



Obr. 4.9 - Diagram sekvencií pre cestu do XRF

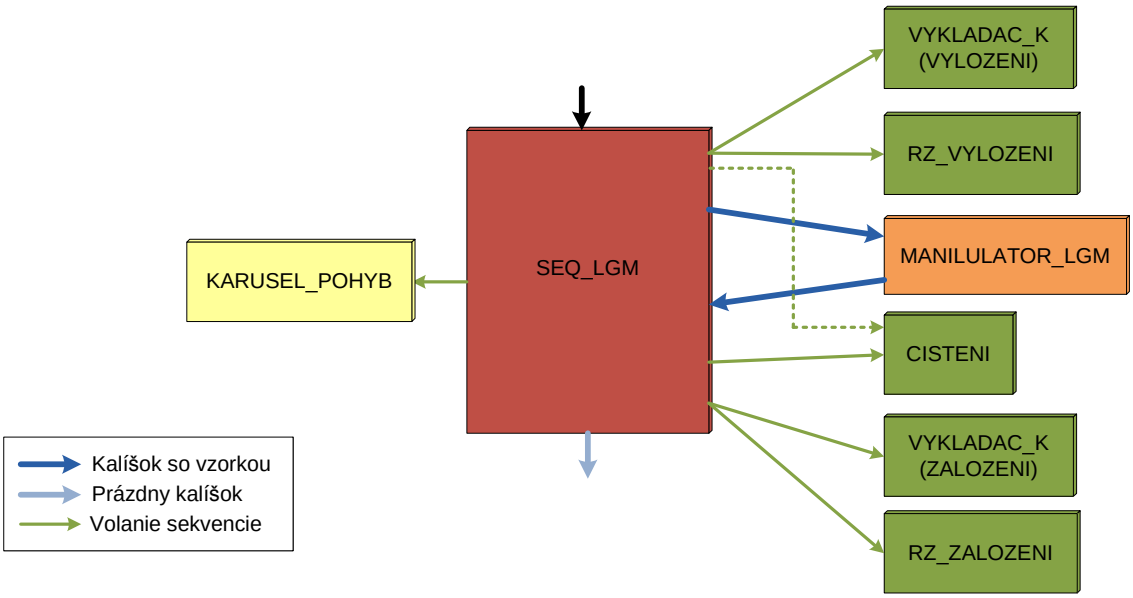
Po vyložení sa vysunie zarážka pred mlynom a spustia sa pásy 1 a 2, takže kalíšok sa dopraví pred mlyn. Tu na seba obidve synchrónne vetvy čakajú a po ich dokončení a splnení podmienky pripraveného odprašovania sekvencia pošle povel mlynu, ktorý si kalíšok prevezme a spustí program mletia. Keď mlyn dokončí mletie, sekvencia predá kalíšok nasledujúcej sekvencii DO_LISU. Po potvrdení prevzatia sa sekvencia DO_MLYNKU vráti do stavu čakania a je pripravená prevziať ďalší kalíšok.

Sekvencia DO_LISU dá po prevzatí kalíšku povel mlynu na jeho vyloženie a následne sa tiež rozdelí na dva synchrónne vlákna. Tak isto ako v predchádzajúcom prípade prvé slúži na prepravu kalíšku, druhé na načítanie parametrov pre lis. Po vyložení kalíšku sa spustí pás 2, ktorý dopraví kalíšok pred lis. Po správnom naparametrovaní lisu a pripravenom odprášení dá sekvencia povel lisu na prevzatie kalíšku a štart programu lisovania. Po vyprázdnení kalíšku nastane rozdelenie vzorku a kalíšku. Kalíšok prevezme sekvencia Z_LISU, vzorku po jej nalisovaní do krúžku sekvencia XRF. Sekvencia DO_LISU sa vráti na začiatok a je pripravená prevziať ďalší kalíšok.

Sekvencia Z_LISU pošle lisu povel na vyloženie kalíšku a spustí pás 2. Po dorazení kalíšku na snímač na konci pásu 2 sa zastaví a čaká na voľný pás 1. Po jeho uvoľnení sa v prípade cesty do RZ vysunie zarážka. Následne nasunovač potlačí kalíšok na pás 1 a kalíšok sa dopraví pred RZ a pomocou sekvencie RZ_ZALOZENI sa dostane na zberný pás. V prípade cesty do karuselu sa kalíšok zastaví na konci pásu 1, kde čaká na uvoľnenie karuselu a jeho následné natočenie na správnu pozíciu. Potom je sekvenciou VYKLADAC_K založený naspäť, natočený na pozíciu čistenia a spustením sekvencie CISTENI vyčistený. Týmto je práca s kalíškom ukončená, je označený za prázdny a pripravený na novú vzorku.

Sekvencia XRF sa stará o vyloženie krúžku so vzorkou z lisu a jeho dopravu do röntgenu, ako aj o jeho cestu naspäť. V lise sa nachádza viac krúžkov, ktoré sa môžu v röntgene uložiť a postupne analyzovať, takže sekvencia musí byť schopná z počiatočného stavu prijať príkaz na ktorúkoľvek z týchto ciest. Ďalšou požiadavkou je obsluha dvoch röntgenov, pričom ich prepínanie je aktuálne riešené navolením z vizualizácie. Sekvenciu som ale pre budúcnosť pripravil tak, aby bola schopná posielať vzorky do ktoréhokoľvek röntgenu. Program okrem prepravy krúžkov musí zaobstarať aj prevzatie vzorku pri jeho oddelení od kalíšku v lise. Pretože táto situácia môže nastať v priebehu prepravy iného krúžku, musí obsahovať dva súčasne bežiace vlákna. Sekvencia sa teda hneď po spustení rozdelí na dva synchrónne vlákna, ktoré sa ale v tomto prípade počas normálneho behu nesynchronizujú, ale neustále bežia nezávisle na sebe v cykloch. Prvé vlákno slúži na prevzatie vzorku a jeho zapamätanie, druhé má na starosti samotnú prepravu krúžku a reaguje na povely od röntgenov alebo lisu. Pri preprave krúžku z lisu do röntgenu si prevezme vzorku zapamätanú prvým vláknom, a pracuje s ňou, pričom prvé vlákno už môže prevziať ďalšiu. V sekvencii XRF sa teda môžu nachádzať dva vzorky, viac nie je možné z princípu funkcie lisu, pretože ten nepovolí vloženie nového kalíšku skôr, ako sa pôvodný krúžok vyloží.

Postup sekvencií pre cestu do LGM je na obr. 4.10. Na rozdiel od cesty do XRF sa tu môže nachádzať iba jeden kalíšok, pretože ide rovno do LGM bez akéhokoľvek predchádzajúceho spracovania. Po prijatí príkazu sa sekvencia rovnako ako v prípade cesty do XRF rozdelí podľa zdroja kalíšku, opäť sa v prípade karuselu kontroluje dostatočné množstvo vzorku. Ak je vzorku dostatok, kalíšok sa vyloží na pás, vysunie sa zarážka pred LGM a spustí sa pás 1.

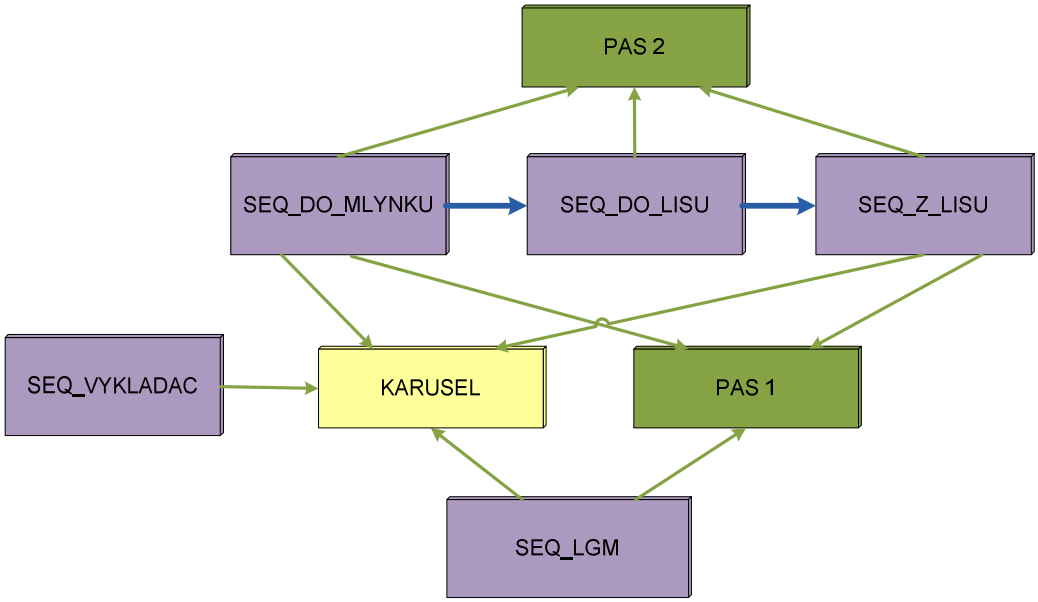


Obr. 4.10 - Diagram sekvencií pre cestu do LGM

Po dorazení na zarážku nadradená sekvencia LGM aktivuje sekvenciu MANIPULATOR_LGM, ktorá kalíšok pomocou pneumatickej ruky uchopí a presunie nad násypku LGM. V tejto pozícii pošle LGM povel na prípravu na analýzu a čaká na potvrdenie a možnosť odsávania. Pokiaľ sú tieto podmienky splnené, sekvencia pokračuje vysypaním kalíšku do násypky a poslaním príkazu na začatia analýzy do LGM. Po tomto kroku čaká na uvoľnenie pásu 1, pretože počas tejto doby mohol byť pás použitý pre prepravu kalíšku do mlynu alebo z lisu. V prípade voľného pásu kalíšok vráti na pás a sekvencia MANIPULATOR_LGM tým končí, kalíšok opäť preberá sekvencia LGM. Tá sa počas práce manipulátoru rozdelila na dva synchrónne vlákna, prvé zodpovedné za prepravu kalíšku a druhé postupne zachytáva stavy analýzy vzorku v LGM. Toto vlákno zabezpečuje hlavne to, aby sa neposlal ďalší kalíšok do LGM pred dokončením aktuálnej analýzy a vyčistením, pretože samotný LGM sa tvári ako pripravený na ďalšiu vzorku už počas čistenia. Práve toto niekedy spôsobovalo chybu v starom systéme, kedy bola nová vzorka odsatá z násypky v cykle čistenia po predchádzajúcom meraní. Po položení kalíšku na pás sekvencia pokračuje rovnakým spôsobom ako sekvencia Z_LISU, kalíšok dopraví do RZ alebo karuselu, kde ho aj vyčistí a označí za prázdny.

4.3.6 Súčasný beh hlavných sekvencií a zdieľanie prostriedkov

Hlavné sekvencie pre riadenie pohybu kalíškov môžu byť aktívne všetky súčasne, preto bolo nutné zabezpečiť zdieľanie určitých častí medzi týmito sekvenciami. Ako vidieť z obr. 4.11, najviac sekvencií využíva karusel, ktorý má ešte navyše manuálny režim, ďalej sú zdieľané pásy 1 a 2.



Obr. 4.11 - Schéma zdieľania zariadení

V prípade karuselu jeho zdieľanie a prideľovanie rieši sekvencia KARUSEL_POHYB. Riešenie prístupu sekvencií k pásom je integrované vo všetkých týchto sekvenciách priamo. Každá sekvencia obsahuje príslušné vstupy POVOLENI_PASx_IN a výstupy POVOLENI_PASx_OUT. Do vstupu pritom vedie výsledok bloku AND zo výstupov ostatných sekvencií. Aby mohla sekvencia použiť pás, musí mať teda na príslušnom vstupe jednotku. Toto riešenie sme zvolili z hľadiska bezpečnosti, keď sa niektorá zo sekvencií ukončí neštandardne, nebude ostatným sekvenciám umožnený prístup k pásu. Týmto sa síce zablokuje časť laboratória, je to však bezpečnejšie, pretože sa nepomiešajú kalíšky. Aby sme získali čo najlepšie zrýchlenie prepravy, každý pás sa blokuje samostatne a jeho uvoľnenie nastane čo najrýchlejšie ako kalíšok tento pás opustí. V prípade sekvencie LGM je teda pás 1 blokovaný od vyloženia kalíšku a uvoľnený po jeho zodvihnutí manipulátorom. To umožní využitie tohto pásu pre cestu ďalšieho kalíšku do mlynu alebo naopak z lisu. Ďalšie urýchlenie sme dosiahli rozdelením cesty z lisu na dve časti, kalíšok je z lisu vyložený a po páse 2 presúvaný aj v prípade, ak nie je voľný pás 1. Na konci pásu 2 totiž môže tento kalíšok počkať na uvoľnenie prvého pásu a potom hneď pokračovať v ceste na páse 1.

4.3.7 Karusel

Ku karuselu patria dve samostatné časti programu. Prvá sa stará o zisťovanie, zapamätanie a zmeny stavu jednotlivých kalíškov a nádob, druhá o riadenie jeho pohybu na správnu pozíciu. Prvá časť je riešená ako vlastný funkčný blok FB KARUSEL_STATUS, druhá je riešená sekvenciou KARUSEL_POHYB v SFC.

Stav karuselu

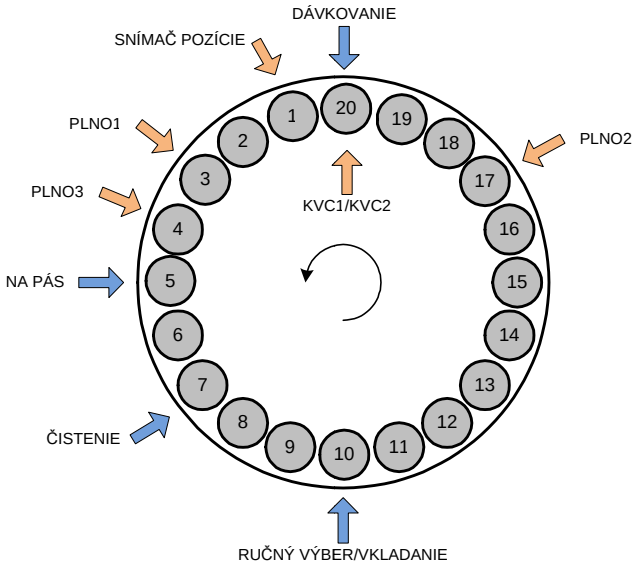
Udržiavanie a aktualizácia stavu jednotlivých kalíškov a nádob je využívaná ostatnými časťami programu, ide hlavne o kontrolu prítomnosti prázdneho kalíšku na pozícii zo strany vzorkovačky. Tak isto je tento blok použitý pre vizualizáciu, aby mali operátori prehľad čo sa s karuselom deje. Z dôvodu potreby zapamätania si informácií sme zvolili funkčný blok, ktorý má na rozdiel od funkcie statické premenné aj výstupy uchovávané v jeho inštančnom DB. Karusel má 20 pozícií, pričom nepárne (liché) sú kalíšky a párne (sudé) sú nádoby na dlhodobý zber vzoriek. Informácie o stave jednotlivých kalíškov sa uchováva vo výstupnej dátovej štruktúre STATUS, ktorá pozostáva z dvadsiatich rovnakých štruktúr CUP_STATUS (tab. 4.1).

Názov	Dátový typ	Komentár
PRESENT	BOOL	Prítomný kalíšek/nádoba
NOT_EMPTY	BOOL	Obsahuje materiál (nemusí byť plný)
FULL_LGM	BOOL	V kalíšku je dostatok vzorku pre analýzu v LGM - snímač PLNO1
UNKNOWN	BOOL	Neznámy stav kalíšku
IN_OPERATION	BOOL	Na pozícii prebieha operácia (dávkovanie, čistenie ...)
FULL_MILL	BOOL	V kalíšku je dostatok vzorku pre analýzu v XRF - snímač PLNO3
rez_1	BOOL	rezerva
CONTAINER	BOOL	Označuje pozíciu pre nádobu (párna), nemení sa

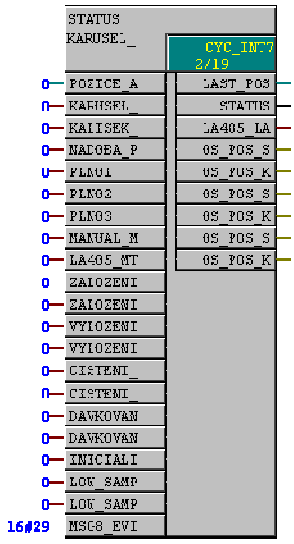
Tab. 4.1 - Štruktúra CUP_STATUS

V tejto štruktúre sú uchované informácie o stavoch všetkých pozícií v poradí od 1 do 20 a využíva sa pre potreby ďalších blokov ktoré potrebujú tieto informácie. Stavy sú aktualizované pri každej zmene natočenia karuselu, pri dokončení nejakej operácie na pozícii (dávkovanie, čistenie, vykladanie...) a pri manuálnom odobraní alebo pridaní kalíšku.

Umiestnenie snímačov a pozícií na ktorých sa vykonávajú jednotlivé operácie sú znázornené na Obr. 4.12. Na pozícii DÁVKOVANIE sa do kalíšku vysype požadované množstvo materiálu. Na tejto pozícii sa tiež kontroluje prítomnosť kalíškov a nádoby, preto sa nemôže stať, že by sa materiál nasypal na prázdnu pozíciu. Pri otáčaní karuselu sa kalíšok dostane na pozíciu PLNO1 a PLNO3, čo sú snímače dostatočného množstva materiálu pre analýzu v LGM a XRF. Snímač PLNO3 v starom systéme nebol, pridali sme ho dodatočne pre kontrolu kalíškov určených do XRF, pretože keď mlyn zistí nedostatok materiálu, musí sa z neho kalíšok ručne vybrať. Takto sa zistí nedostatok materiálu už v karuseli a systém ho automaticky pošle na čistenie bez zásahu operátora a označí vzorku za chybnú.



Obr. 4.12 - Schéma snímačov a pozícií na karuseli



Obr. 4.13 - Blok KARUSEL_STATUS

V pozícii NA PÁS sa vykladá kalíšok na pás (resp. zakladá z pásu). Tu sa už využíva zapamätaná informácia o prítomnosti kalíšku, pokiaľ sa stane že niekto kalíšok vyberie z karuselu počas jeho pohybu medzi snímačom KVC1 a pozíciou vykladania, systém to nemá ako zistiť. Pokiaľ sa tak stane, systém nebude o neprítomnosti kalíšku vedieť a zastaví sa až na maximálnom čase čakania na senzor prítomnosti kalíšku na páse. Táto situácia by však nemala nastať, pretože do karuselu by nemal nikto zasahovať inak, ako pomocou na to určených dvierok na pozícii ručného výberu/vkladania. Pokiaľ sa kalíšok odoberie alebo pridá na tejto pozícii v manuálnom móde s použitím príslušného tlačítka, systém si informácie aktualizuje a k žiadnej chybe nedôjde. Zaplnenie nádoby sa kontroluje snímačom PLNO2, na rozdiel od kalíškov, kde je ich zaplnenie žiadané, zistenie plnej nádoby znamená nutnosť výmeny za prázdnu a preto generuje chybu a hlášku na OS.

Na úpravu stavu pre žiadanú pozíciu sme použili nepriame adresovanie, pre ktoré treba vždy najprv vypočítať ukazateľ (pointer) na adresu. Výpočet je nasledovný:

$$MOD_{20}\ (\text{aktuálna pozícia} + \text{posun od snímača-1}) * 16 + \text{P\#\#STATUS}$$

Aktuálna pozícia s posunom je číslo kalíšku na snímači, jednotku odčítame z dôvodu adresovania od nuly. MOD_{20} zabezpečí prechod cez maximálnu pozíciu 20 (rozsah 0 .. 19). Výsledok je vynásobený dĺžkou jednej štruktúry CUP_STATUS (16 bitov) a pripočítaný k ukazateľu na štruktúru STATUS. V kóde STL potom výpočet pre posun 19 vyzerá nasledovne:

```
      L     #POZICE_AKT
      L     18
      +D
      L     20
      MOD
      L     16
      *D
      L     P##STATUS
      +D
      LAR1
```

Po tomto výpočte sa v AR1 nachádza ukazateľ na štruktúru CUP_STATUS pozície na snímači prítomnosti KVC1/KVC2 a program pokračuje kontrolou stavu, napríklad pre prítomnosť kalíšku na tejto pozícii je nasledujúci kód:

```
KAL2: A     #KALISEK_POZ        // SNIMAC PRITOMNOSTI KALISKU
      XN    [AR1,P#0.0]
      JC    MAN                 // POKRACUJE AK JE PAMATANY STAV
                                       ROVNAKY AKO DETEKOVANY
      A     #KALISEK_POZ
      JC    KPR1

// NENI PRITOMNY - NASTAVI STAV NA NEPRITOMNY A UNKNOWN + ALARM 2
      SET
      S     #ALA02_CUP_MISSING
      L     8
      T     B [AR1,P#0.0]
      JU    MAN

//JE PRITOMNY - NASTAVI NA PRAZDNY KALISEK, BEZ ALARMU
KPR1: S      [AR1,P#0.0]
      L     1
      T     B [AR1,P#0.0]
      JU    MAN
```

Kontrola na ostatných snímačoch je podobná, líši sa hlavne v nastavovaných hodnotách do štruktúry CUP_STATUS, prípadne nastaveniach alarmov. Ďalšie zmeny stavov sa dejú na pozíciách operácií s kalíškom alebo nádobou (na obr. 4.12 označené modrými šípkami), v tomto prípade sa na správnej pozícii nastavia príslušné bity podľa povelu.

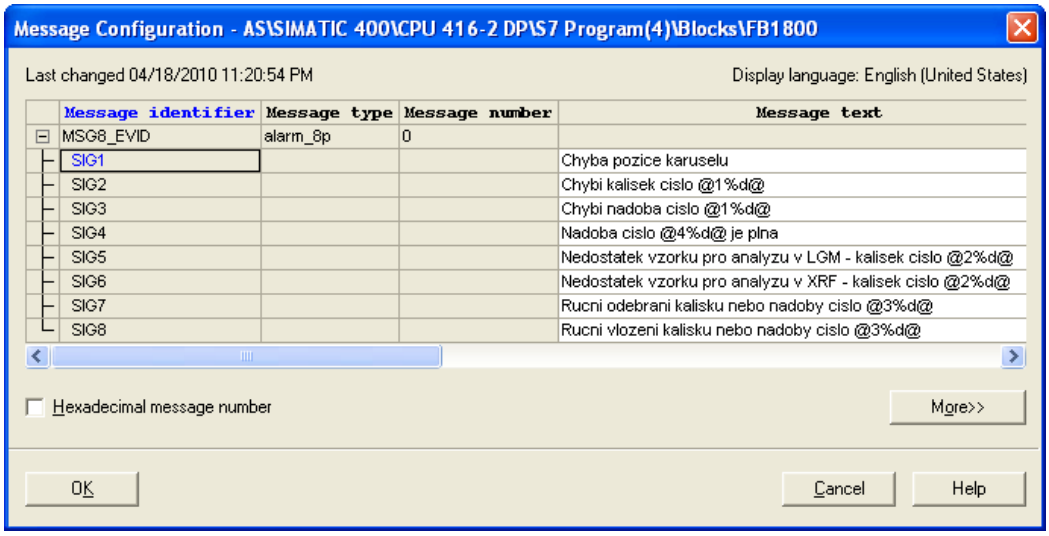
Aby bolo možné vizualizovať karusel správne, musia sa pozície presúvať, čo by bol vo WinCC dosť zložitý problém a tiež by to celú vizualizáciu spomaľovalo. Jednoduchší spôsob je však tento presun urobiť priamo v FB a vo vizualizácii zobrazovať už posunuté stavy. Pre túto potrebu má FB ďalších dvadsať výstupov, ktoré predstavujú stavy jednotlivých pozícií karuselu (OS_POS_ST_x). Tie obsahujú také isté stavy ako štruktúra STATUS (posunuté) ale sú typu BYTE, pretože zo štruktúry sa nedá vygenerovať tag pre WinCC a teda ani podľa nej dynamizovať objekty vo vizualizácii. Pre lepšiu prehľadnosť som vytvoril aj výstupy s číslom kalíšku pre vizualizáciu (OS_POS_K_x). Tieto pozície sú vždy posunuté o aktuálne natočenie karuselu. Opäť som použil nepriame adresovanie, tentokrát sú nepriamo adresované zdrojové aj cieľové dáta v cykle.

```
OSUP: L     #LAST_POSITION         // POZICIA KARUSELU
      T     #POSUN_OUT
      L     0
      T     #POSUN_ST
OSL:  L     #POSUN_OUT
      L     1
      -D
      L     16
      *D
      L     P##STATUS
      +D
      T     #L_AR3                 // POINTER NA AKTUALNY CUP_STATUS

      L     #POSUN_ST
      L     16
      *D
      L     P##OS_POS_ST_1
      +D
      LAR1                         // POINTER NA POSUNUTU ADRESU OS_POS_ST_x

      L     DIB [#L_AR3]           // KOPIROVANIE STAVU ZO STATUS
      T     B [AR1,P#0.0]          // NA POSUNUTY VYSTUP OS_POS_ST_x
      L     #POSUN_OUT             // KOPIROVANIE CISLA POZICIE
      T     B [AR1,P#1.0]          // NA POSUNUTY VYSTUP OS_POS_K_x

      L     #POSUN_OUT
      INC   1                      // INKREMENTOVANIE POSUNU
      L     20
      MOD
      JN    OS20
      L     20
OS20: T     #POSUN_OUT             // ZAISTENIE ROZSAHU POSUNU 1 .. 20
      L     #POSUN_ST
      INC   1
      T     #POSUN_ST
      L     20
      <I
      JC    OSL                    // CYKLUS KYM V POSUN_ST NIE JE 0
```

Na konci bloku je volanie SFB35 ALARM8P, čo je blok umožňujúci po správnom nakonfigurovaní vygenerovať alarm do WinCC a predať do nej parametre. Blok dokáže vygenerovať 8 rôznych alarmov s desiatimi hodnotami. V tomto prípade sú parametre vždy čísla kalíškov, aby bolo jednoduchšie určiť zdroj chyby.

```
AL8P:   CALL   #ALARM8P
        EN_R      :=TRUE
        SIG_1     :=#ALA01_POSITION
        SIG_2     :=#ALA02_CUP_MISSING
        SIG_3     :=#ALA03_CONT_MISSING
        SIG_4     :=#ALA04_CONT_FULL
        SIG_5     :=#ALA05_LOW_SAMPLE_LGM
        SIG_6     :=#ALA06_LOW_SAMPLE_MILL
        SIG_7     :=#ALA07_MANUAL_REMOVE
        SIG_8     :=#ALA08_MANUAL_PUT_ON
        ID        :=W#16#EEEE
        EV_ID     :=#MSG8_EVID
        SEVERITY  :=W#16#40
        DONE      :=#ALARM_DONE
        ERROR     :=#ALARM_ERROR
        STATUS    :=#ALARM_STATUS
        ACK_STATE:=#ALARM_ACK_STATE
        SD_1      :=#OS_POS_K_20        // CISLO NA POZICII DAVKOVANIA
        SD_2      :=#OS_POS_K_5         // CISLO KALISKU NA POZICII VYKLADANIA
        SD_3      :=#OS_POS_K_10        // CISLO NA POZICI RUCNEHO VKLADANI
        SD_4      :=#OS_POS_K_17        // CISLO NA POZICII ZAPLNENIA NADOBY

        CLR                             // VYNULOVANIE CHYBOVYCH BITOV
        =      #ALA05_LOW_SAMPLE_LGM
        =      #ALA06_LOW_SAMPLE_MILL
        =      #ALA07_MANUAL_REMOVE
        =      #ALA08_MANUAL_PUT_ON
```

Text hlásení sa definuje v Message Configuratore, parametre sa vkladajú pomocou @[CisloParametru] [Format]@. Nakonfigurované hlásenia sú na obr. 4.14



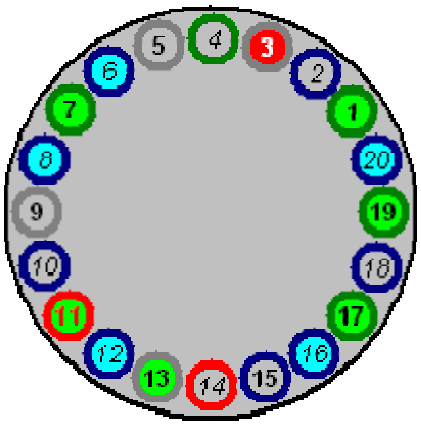
Obr. 4.14 - Konfigurácia textov hlásení bloku KARUSEL_STATUS

Na vizualizáciu stavu jednotlivých kalíškov sme navrhli farebné zobrazenie, pričom stavy ktoré existovali v starom systéme sme ponechali z dôvodu jednoduchšieho prechodu pre operátorov. Všetky použité stavy sú zobrazené v tab. 4.2.

	Kalíšky	Nádoby
Prázdna pozícia	15	16
Prázdny kalíšok/nádoba	17	18
Obsahuje materiál	1	2
Plný kalíšok/nádoba	3	4
Na pozícii prebieha operácia	19	20
Chyba - nekorektné odobranie	11	12
Chyba - nedostatok vzorku na analýzu	1	

Tab. 4.2 - Stavy kalíškov a nádob

Každá pozícia je vizualizovaná jedným Customized Objectom, ktorý je spojený s jemu prislúchajúcimi výstupmi OS_POS_K_x a OS_POS_ST_x. Hodnota výstupu OS_POS_K_x predstavuje číslo kalíšku alebo nádoby prítomnej na zobrazovanej pozícii, takže sa priamo zobrazí. Hodnota a OS_POS_ST_x je stav pozície, z ktorého sa pomocou C-Scriptu menia farby podľa navrhnutého štandardu. Pre lepšie odlíšenie sú čísla kalíškov zobrazované tučným písmom, čísla nádob kurzívou. Customized object som vytvoril tak, aby jeho vlastnosti ktoré chcem skriptom meniť mali štandardný tvar a boli previazané s požadovanými vlastnosťami jednotlivých objektov z ktorých je zložený. Preto farba popredia a pozadia objektu je naviazaná na objekt kruhu, ale farba a typ písma na objekt textu s číslom kalíšku. Výsledný vzhľad vizualizácie karuselu s rôznymi stavmi pozícií (simulovanými) je na obr. 4.15.



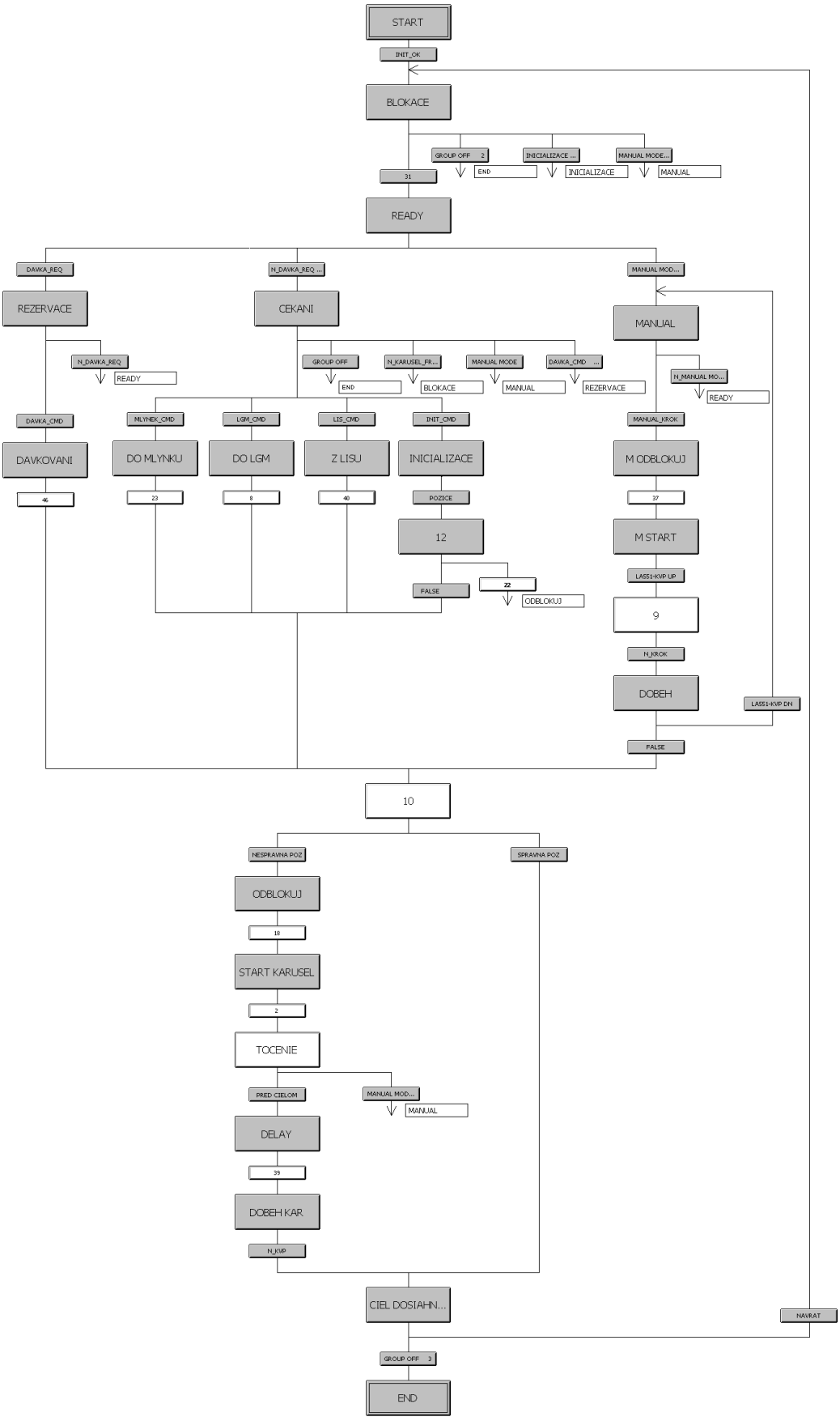
Obr. 4.15 - Príklad vizualizácie karuselu

Pohyb karuselu

Pohyb karuselu je riadený sekvenciou vytvorenou v SFC (obr. 4.16), ktorá zabezpečuje pohyb karuselu v automatickom alebo manuálnom móde a tiež jeho inicializáciu. Inicializovať karusel je nutné hlavne po nahrávaní programu PLC, prípadne pri chybe karuselu kedy môže zostať stáť v medzipolohe. Vďaka tomu už nie je v takomto prípade nutné ísť ku karuselu a manuálne ho natočiť do niektorej z pozícií. Manuálny režim sa využíva hlavne v prípade poruchy, kedy je treba s karuselom točiť z miesta pomocou tlačítka. Do manuálneho režimu sa dostane prepínačom umiestneným priamo na karuseli, po jeho prepnutí sekvencia prejde do stavu MANUAL, v ktorom čaká na stlačenie tlačítka. V tomto stave samozrejme neprijíma požiadavky od ostatných sekvencií, v manuálnom režime je neprípustné samovoľné otočenie karuselu. Manuálny režim sa tiež využíva na ručné odoberanie a vkladanie kalíškov a nádob, tieto operácie je nutné robiť na pozícii na to určenej.

V automatickom móde sa po spustení sekvencie dostane najprv do stavu BLOKACE, odkiaľ po splnení všetkých podmienok prejde do stavu READY. V tomto stave je zabezpečená preferencia natočenia karuselu na pozíciu dávkovania. Sekvencia Vykladač nastaví požiadavku DAVK_REQ, čo zabezpečí že zo stavu READY sekvencia prejde do stavu REZERVACE v ktorej čaká na povel od sekvencie Vykladača. Ostatné sekvencie v tomto prípade nedostanú signál Ready, preto čakajú. V prípade bez rezervácie vykladačom sa sekvencia dostane do stavu CEKANI. V tomto stave je karusel voľný pre všetky sekvencie. Samozrejme z tohto stavu je opäť možné prejsť do stavu REZERVACE pre vykladač. Po príchode povelu na natočenie od sekvencie sa zapamätá žiadaná pozícia a pokiaľ je rôzna od aktuálnej, pokračuje sekvencia do vetvy hľadania žiadanej pozície.

Karusel musí zastaviť vždy na danej pozícii presne, preto obsahuje okrem pohonu a snímačov polohy aj signál Blokace. Pomocou tohto signálu je možné zastaviť ho presne na správnom mieste, je ale potrebné ho zablokovať v správnom čase. Ideálne je to tesne po odchode z predchádzajúcej pozície, ale nie príliš skoro, aby sa karusel nezablokoval ešte na nej. Sekvencia riadenia pohybu teda najprv odblokuje karusel, v ďalšom kroku spustí pohon a v stave TOCENIE sníma aktuálnu pozíciu karuselu. Po príchode na pozíciu predchádzajúcu žiadanej prejde ďalej, kde krok DELAY zabezpečuje oneskorenie (odladené na 500 ms), po ktorom sa karusel zablokuje, ale pohon stále beží, až kým nepríde signál správnej pozície. V tomto stave je karusel zastavený na presnej pozícii, aj keby bol signál na pohyb prítomný o niečo dlhšie. Po dokončení natáčania sa v kroku CIEL_DOSIAHNUTY nastaví uvoľnenie pre ostatné sekvencie a stav prejde opäť na začiatok.



Obr. 4.16 - Sekvencia KARUSEL_POHYB

4.3.8 Posun času

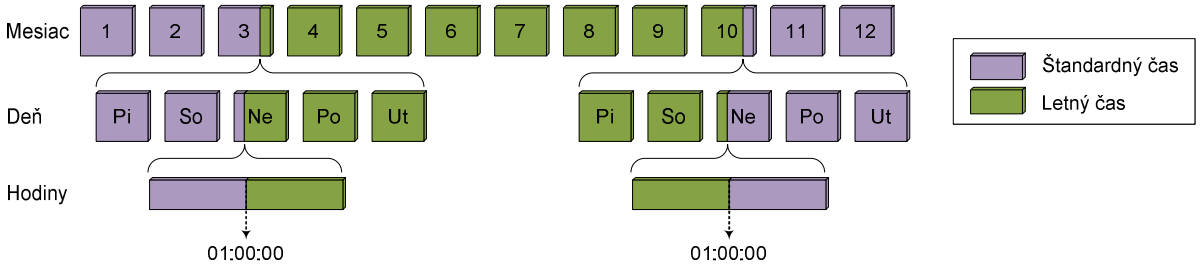
PLC pracuje štandardne s časom UTC, čo je výhodné, pretože sa nemusia riešiť problémy s prechodom na letný čas. V prípade laboratória, kde je dôležité poznať čas analýzy vzorku, je potrebné tento čas prispôsobiť aktuálnemu času, ktorý používajú ľudia. Tak isto časy odberu pravidelných vzoriek z jednotlivých vzorkovačiek sú stanovené na určité hodiny, takže bez vyriešenia posunu času na letný by sa všetko posunulo o hodinu, čo by spôsobovalo problémy pri orientácii vo výsledkoch a tvorbe reportov. Takáto funkcia však v systéme PCS 7 nie je, práve z dôvodu že obvykle je UTC výhodnejší, takže som ju musel vytvoriť.

Počas štandardného času je v ČR posun proti UTC +1 hodina, počas letného +2 hodiny. Zmena času na letný sa deje vždy poslednú nedeľu v marci (březen) o 00:59:59 UTC, takže sa posunie čas z 1:59:59 na 3:00:00. Letný čas končí v poslednú nedeľu v októbri (říjen), tak isto o 00:59:59 UTC, a čas sa posunie z 2:59:59 na 2:00:00 [4]. Funkciu som navrhol univerzálne tak, aby sa dala použiť aj v iných časových pásmach, takže jej jediný vstupný parameter je posun štandardného času voči UTC, výstup je posun aktuálneho času.

Na začiatku funkcie je nutné zistiť aktuálny čas. Na to slúži SFC1 READ_CLK, ktorej výsledkom je aktuálny dátum a čas vo formáte Date_And_Time. Tento formát je výhodný pretože obsahuje dátum, čas aj deň v týždni.

```
CALL  "READ_CLK"                    // NACITANIE AKTUALNEHO CASU
 RET_VAL:=#READ_CLK_ERR
 CDT    :=#DT_ACTUAL
```

Pri kontrole postupuje funkcia tak ako je naznačené na obr. 4.17, najprv po mesiacoch, ak je celý mesiac v jednom čase, funkcia nastaví výsledný posun okamžite a pokračuje na záver funkcie, kde sa ešte pripočíta posun voči UTC. V prípade mesiacov marec a október, v ktorých dochádza k zmene času je funkcia zložitejšia, pretože musí zistiť, či je aktuálny čas pred alebo po poslednej nedeli v mesiaci.



Obr. 4.17 - Štandardný a zimný čas počas roka

Pre túto kontrolu už bude potrebné poznať aktuálny deň, hodinu a deň v týždni. Tu sa prejaví výhoda formátu Date_And_Time, pretože vypočítať deň v týždni tiež nie je jednoduché.

```
LAR1  P##DT_ACTUAL
L     B [AR1,P#2.0]
BTI
T     #DAY                        // VYBER DNA Z AKTUALNEHO DT
L     B [AR1,P#3.0]
BTI
T     #HOURS                      // VYBER HODIN Z AKTUALNEHO DT
L     B [AR1,P#7.0]
L     B#16#F
AW
BTI
T     #DAY_OF_WEEK                // VYBER DNA V TYZDNI Z AKTUALNEHO DT
```

Pretože marec aj október majú 31 dní, posledná nedeľa bude mať určite dátum minimálne 25 a teda dni pred 25. sú automaticky pred zmenou. Posledný test týkajúci sa celých dní je zistenie hodnoty DAY - DAY_OF_WEEK, ak je táto hodnota vyššia alebo rovná 24, je aktuálny deň súčasťou posledného týždňa.

```
L     #DAY
L     25
<I
JC    BEF                         // DEN < 25 => PRED POSLEDNOU NEDELOU

L     #DAY
L     #DAY_OF_WEEK
-I
L     24
<I
JC    BEF                         // PRE POSLEDNY TYZDEN V MESIACI
                                  // PLATI (DEN - DEN V TYZDNI) >=24
```

Keď je DAY_OF_WEEK vyšší ako 1 (nedeľa je podľa štandardu prvý deň v týždni), je tento deň už po zmene času. Funkcia teda pokračuje iba v prípade, že DAY_OF_WEEK sa rovná 1. Tu už stačí vyhodnocovať hodiny, pred dosiahnutím hodnoty 1 je pred zmenou, po jej dosiahnutí je po zmene.

```
L     #DAY_OF_WEEK
L     1                           // NEDELA JE PRVY DEN V TYZDNI !!!
<>I
JC    AFT                         // PO NEDELi V POSLEDNOM TYZDNI
L     #HOURS
L     1
>=I
JC    AFT                         // AKTUALNA HODINA >=1 - PO ZMENE
```

Tým je v akomkoľvek okamihu presne definované, či je aktuálny čas pred zmenou alebo po zmene. Z tejto informácie je však nutné určiť podľa aktuálneho mesiaca aký čas nastaviť.

```
BEF:  NOP   0                          // PRED ZMENOU
      L     #MONTH
      L     3
      ==I
      JC    WINT                       // MAREC - NASTAV STANDARDNY CAS
      JU    SUMM                       // INAK (OKTOBER) NASTAV LETNY

AFT:  NOP   0                          // PO ZMENE
      L     #MONTH
      L     3
      ==I
      JC    SUMM                       // MAREC - NASTAV LETNY CAS
      JU    WINT                       // INAK (OKTOBER) NASTAV STANDARDNY
```

Nakoniec k výsledku pripočítanie časového posunu voči UTC a nastavenie výslednej hodnoty.

```
WINT: L     #UTC_SHIFT
      T     #RET_VAL
      JU    END

SUMM: L     #UTC_SHIFT
      L     T#1H
      +D
      T     #RET_VAL
      JU    END
```

Funkciu som otestoval na niekoľko rokov dopredu, a pretože sa tento systém nasadzoval a odladďoval aj v priebehu marca, otestoval som ju aj priamo v nasadení.

4.3.9 Sekvencia na nastavenie parametrov pre mlyn a lis

Aby mal materiál vlisovaný v krúžku správne parametre a nedošlo k jeho vysypaniu, musí byť správne pomletý a zlisovaný. Pre každý analyzovaný materiál je teda potrebné nastaviť požadované parametre mletia a lisovania. Tieto parametre sa nastavujú pre každý typ materiálu vo webovej aplikácii CEMLAB a uchovávajú sa v databáze. Pred vložením kalíšku do mlynu alebo lisu je teda nutné tieto parametre načítať z databázy a poslať ich cez komunikačný blok do mlynu (resp. lisu). Pretože tieto prenosy niekedy prebehnú s chybou, bolo potrebné vytvoriť sekvenciu s možnosťou opakovania týchto pokusov. Vytvoril som ju tak, aby sa dal v prípade potreby zmeniť počet opakovaní. Po prijatí povelu od nadradenej sekvencie sa pokúsi získať parametre, po ich prijatí a potvrdení ich prepošle komunikačnému bloku mlynu alebo lisu. Pokiaľ sa niektorý z týchto prenosov nepodarí uskutočniť ani po maximálnom opakovaní, sekvencia prejde do chybového stavu. Pri testovaní systému sa nám nikdy nestalo aby sa prenos nepodaril najneskôr na druhý pokus, takže sme počet opakovaní nechali nastavený na tri.

4.3.10 Komunikácia so zariadeniami

Tak ako v projekte Astana, aj v tomto projekte sme narazili na nutnosť komunikovať s inými zariadeniami. Situáciu zjednodušilo to, že v prípade komunikácie s mlynom a lisom sa dal použiť blok U_EXT_DEVICE vytvorený pre tento projekt. Komunikácia s ostatnými zariadeniami, teda LGM a oboma rentgenmi prebieha cez OPC server, takže samotné dáta sú zapisované a čítané z DB vyhradeného pre túto komunikáciu. Pretože zloženie dát je aj pre tieto zariadenia podobné ako v predchádzajúcich prípadoch, vytvorili sme nový blok U_EXT_DEVICE_DB, ktorý z väčšej časti vychádza z bloku U_EXT_DEVICE. Hlavné rozdiely sú už v spomínanej komunikácii cez DB, pridanie výstupov na zobrazovanie času a dátumu a zmeny číselných vstupov a výstupov z REAL na DINT. K tejto zmene sme pristúpili z dôvodu nepoužívania typu real pre tieto zariadenia, a práve potreby prenosu DINT čísel, hlavne ID vzoriek, ktoré by pri sa do typu REAL nezmestili s dostatočnou presnosťou. To by pri konverzii na DINT mohlo spôsobiť stratu poslednej číslice, čo je pri ID neprípustné. Ďalšou zmenou je parametrizovanie bloku, ktoré je pre všetky bloky pristupujúce k tomuto komunikačnému bloku riešené spoločne, parametre sú teda prebrané cez dve štruktúry COM_PARAM a DB_DATA (tab. 4.3).

COM_PARAM			
	DB_NR	Int	Číslo komunikačného DB
	SAMPLE_TIME	Real	SampleTime pre všetky komunikačné bloky
	Comm_Pause	Real	Pauza medzi odoslanim dat a potvrdenim
DB_DATA			
	START_BYTE	Int	Počiatočný byte pre zariadenie v komunikačnom DB
	BYTE_LONG_REC	Int	Počet prijatých bytov
	BYTE_LONG_SEND	Int	Počet odosielaných bytov

Tab. 4.3 - Parametrizačné štruktúry komunikácie

Prijaté dáta tentokrát nebudú pochádzať z komunikačnej funkcie, ale budeme ich do statických dát z komunikačného DB. Na to použijeme funkciu SFC20 BLKMOV, ktorá využíva ako zdroj a cieľ ANY Pointre, takže budeme potrebovať dva - Source a Destination. Pretože v FB je použitie AR2 dosť problematické, umiestnili sme tieto ANY Pointre do lokálnych dát za sebou, takže bude stačiť odkaz na prvý, druhý bude mať vždy adresu o 10 vyššiu, čím sa nám táto časť kódu výrazne zjednoduší. V komunikačnom DB sú najprv dáta na odoslanie, až za nimi sú prijaté dáta. Preto začiatok týchto dát je na adrese vypočítanej ako DB_DATA.START_BYTE + DB_DATA.BYTE_LONG_SEND.

```
LAR1  P##Source
 L     W#16#1002                    // S7 code &   BYTE
 T     W [AR1,P#0.0]
 T     W [AR1,P#10.0]
 L     #DB_DATA.BYTE_LONG_REC       // repetition
 T     W [AR1,P#2.0]
 T     W [AR1,P#12.0]
 L     #COMM_PARAM.DB_NR            // DB number
 T     W [AR1,P#4.0]
 L     0                            // Instancne DB
 T     W [AR1,P#14.0]
 L     P#DBX 0.0
 L     #DB_DATA.START_BYTE          // Zaciatok dat
 SLD   3                            // Konverzia na format pointer
 +D
 L     #DB_DATA.BYTE_LONG_SEND
 SLD   3                            // Konverzia na format pointer
 +D
 T     D [AR1,P#6.0]
 L     P##RECEIVE_DATA              // POINTER
 T     D [AR1,P#16.0]

 CALL  "BLKMOV"
  SRCBLK :=#Source
  RET_VAL:=#RETURN_VALUE
  DSTBLK :=#Destination
```

Po skopírovaní dát do statických nasleduje kopírovanie bitových signálov, ktoré zostalo proti pôvodnému bloku bez zmeny. Následne sa kopírujú číselné hodnoty, princíp z bloku U_EXT_DEVICE zostáva, zmenil sa len spôsob prevodu dátových typov. na prevod z INT na DINT existuje funkcia ITD, prevod z BYTE na DID je jednoducho len prekopírovanie dát z BYTE do DWordu. V poslednom prípade sa tiež jedná len o kopírovanie.

```
CMDI: L     DIW [#P_RecvData]        // Nacitanie hodnoty z prijatych dat
      ITD                            // Prevod INT na DINT
      T     DID [AR1,P#0.0]          // Kopirovanie na aktualny vystup
      L     P#2.0                    // Nacitanie posunu adresy
      JU    CMDC

CMDB: L     DIB [#P_RecvData]        // Nacitanie hodnoty z prijatych dat
      T     DID [AR1,P#0.0]          // Kopirovanie na aktualny vystup
      L     P#1.0                    // Nacitanie posunu adresy
      JU    CMDC

CMDD: L     DID [#P_RecvData]        // Nacitanie hodnoty z prijatych dat
      T     DID [AR1,P#0.0]          // Kopirovanie na aktualny vystup
      L     P#4.0                    // Nacitanie posunu adresy
```

Ďalšie prijaté dáta sú dátumy a časy začiatku a ukončenia analýzy vo formáte DATE_AND_TIME. Tento formát je 8 bytový a neumožňuje jeho priame previazanie s vizualizáciou vo WinCC. Pre tieto potreby sme sa rozhodli prekonvertovať ho na STRING. Formát DATE_AND_TIME obsahuje dátum a čas v BCD kóde a deň v týždni. Dátová štruktúra typu DATE_AND_TIME a STRING[n] je v tab. 4.4.

DATE_AND_TIME									
Byte	0	1	2	3	4	5	6	7 (4 MSB)	7 (4 LSB)
Obsah	Year	Month	Day	Hour	Minute	Second	2 MSD of ms	LSD of ms	Day of week
Rozsah	1990 - 2089	1 - 12	1 - 31	0 - 23	0 - 59	0 - 59	00 - 99	0 - 9	1 - 7
STRING[n]									
Byte	0	1	2	3	4	5	6	...	n + 2
Obsah	max. Dĺžka	akt. Dĺžka	znak 0	znak 1	znak 2	znak 3	znak 4		znak n
Rozsah	n	1 - n	ASCII znaky						

Tab. 4.4 - Dátová štruktúra typov DATE_AND_TIME a STRING

Do vytváraného stringu je teda potrebné zapísať maximálnu a aktuálnu dĺžku, ktoré je v tomto prípade rovnaká a nemenná, pretože string bude dlhý vždy 20 znakov a jeho výsledný formát je :"DD.MM.YYYY HH:MM:SS". Tento STRING[20] má teda spolu 22 bytov, prvé dva budú vždy obsahovať hodnoty 20 (14h)

```
L     W#16#1414                 // length of string
T     LW    8
```

Konverzia hodnôt na string potom prebieha postupne po jednotlivých znakoch, ako príklad uvedieme vytvorenie posledných dvoch číslic roku (prvé dve sú staticky "20"). Prvú číslicu získame z daného bytu z dolných 4 bitov, ostatné vymaskujeme. Aby sme z čísla dostali ASCII kód, treba k hodnote vždy pripočítať konštantu 30h (ASCII kód znaku "0").

```
L     LB    0                   // Nacitanie nulteho bytu DT
L     B#16#F
AW                              // Vymaskovanie prvej cislice
L     B#16#30
+I                              // Prevod na ASCII (+30h)
T     #TMP_DATETIME_STRING[10]  // rok 4. cislica
```

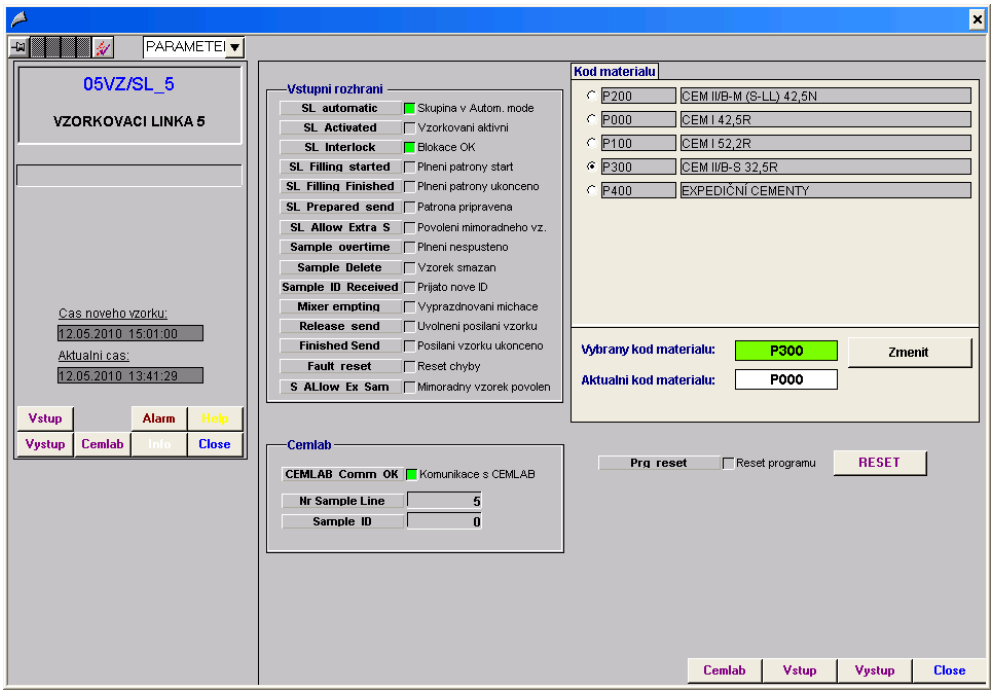
Druhú číslicu získame z horných 4 bitov, ktoré posunom o 4 bity dostaneme na pozíciu dolných 4 bitov a opäť ASCII znak získame pripočítaním 30h.

```
L     LB    0                   // Nacitanie nulteho bytu DT
SRW   4                         // Posun o 4 bity
L     B#16#30
+I                              // Prevod na ASCII (+30h)
T     #TMP_DATETIME_STRING[9]   // rok 3. cislica
```

Takým istým spôsobom získame s ostatných bytov znaky pre zvyšné časti dátumu a času, na koniec ešte pridáme statické oddeľujúce znaky. Takto získaný string potom pomocou funkcie SFC20 BLKMOVE prekopírujeme na príslušný výstup. Tým je hotové spracovanie prijatých dát, následné odoslanie dát je takmer zhodné ako v pôvodnom bloku U_EXT_DEVICE, mierne rozdiely sú opäť iba pri prevode číselných hodnôt na DINT.

4.3.11 Vizualizácia laboratória

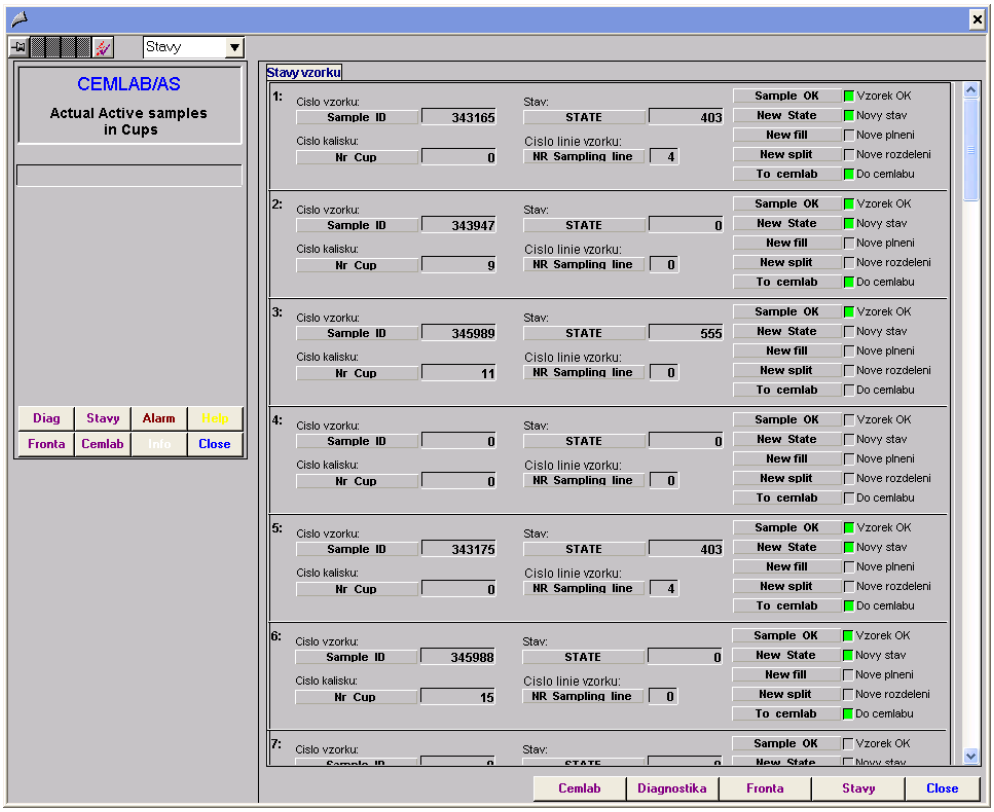
Pri návrhu vizualizácie sme vychádzali z pôvodného rozvrhnutia v starom systéme, aby sme operátorom zjednodušili prechod na novú verziu. Na prehľadovej obrazovke LABORATOR (príloha E) sú zobrazované hlavné údaje o každej vzorkovačke, hlavne aktuálny materiálový kód, dátum a čas poslednej a nasledujúcej vzorky a aktuálny stav vzorkovania. Z tejto obrazovky je možné jednotlivé vzorkovačky spustiť, tak isto ako zobraziť si detailnú obrazovku všetkých vzorkovačiek. Ďalej sme na túto obrazovku umiestnili ikony pre zobrazenie detailov riadiacich blokov jednotlivých vzorkovacích línií (obr. 4.18), ktoré sa používajú hlavne na diagnostiku alebo zmenu materiálového kódu pri prechode na výrobu iného druhu cementu.



Obr. 4.18 - Faceplate bloku SAMPLE_LINE s tabuľkou materiálových kódov

Vedľa jednotlivých vzorkovačiek sú zobrazené všetky odosielacie stanice, ktoré sú s vzorkovačkami prepojené dynamizovanými čiarami zobrazujúcimi pripravenú vzorku. Podobné čiary z odosielacích staníc zobrazujú pripravenosť patróny na odoslanie. Na tieto čiary je následne napojená vizualizácia potrubnej pošty so signalizáciou pohybu patróny a tiež základná vizualizácia najhlavnejších objektov nachádzajúcich sa v laboratóriu. Na pravom okraji obrazovky sú umiestnené ikony skupín všetkých vzorkovačiek a tiež skupiny patriace pod laboratórium. Vďaka tomu je možné z tejto obrazovky vykonávať všetky základné operácie ako spúšťanie a zastavovanie vzorkovania, potrubnej pošty a laboratória.

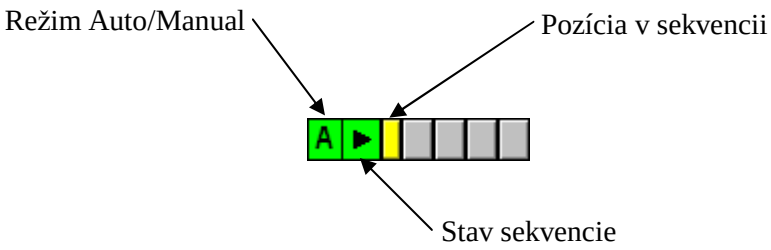
Ďalej sa na prehľadovej obrazovke nachádzajú ikony pre zobrazenie detailov ostatných riadiacich blokov laboratória, ACTUAL_SAMPLE, EXTRA_SAMPLE, SEND_QUEUE a GET_ID. Tieto detailné obrazovky sa pri bežnej prevádzke používajú málo, dôležité sú však pri oživovaní a pri diagnostike prípadných porúch, pretože sú v nich zobrazené stavy všetkých aktívnych vzoriek (obr. 4.19) ako aj fronty odosielania patrón a kalíškov z karuselu.



Obrázok 4.19 - Faceplate bloku ACTUAL_SAMPLE so zoznamom všetkých vzoriek

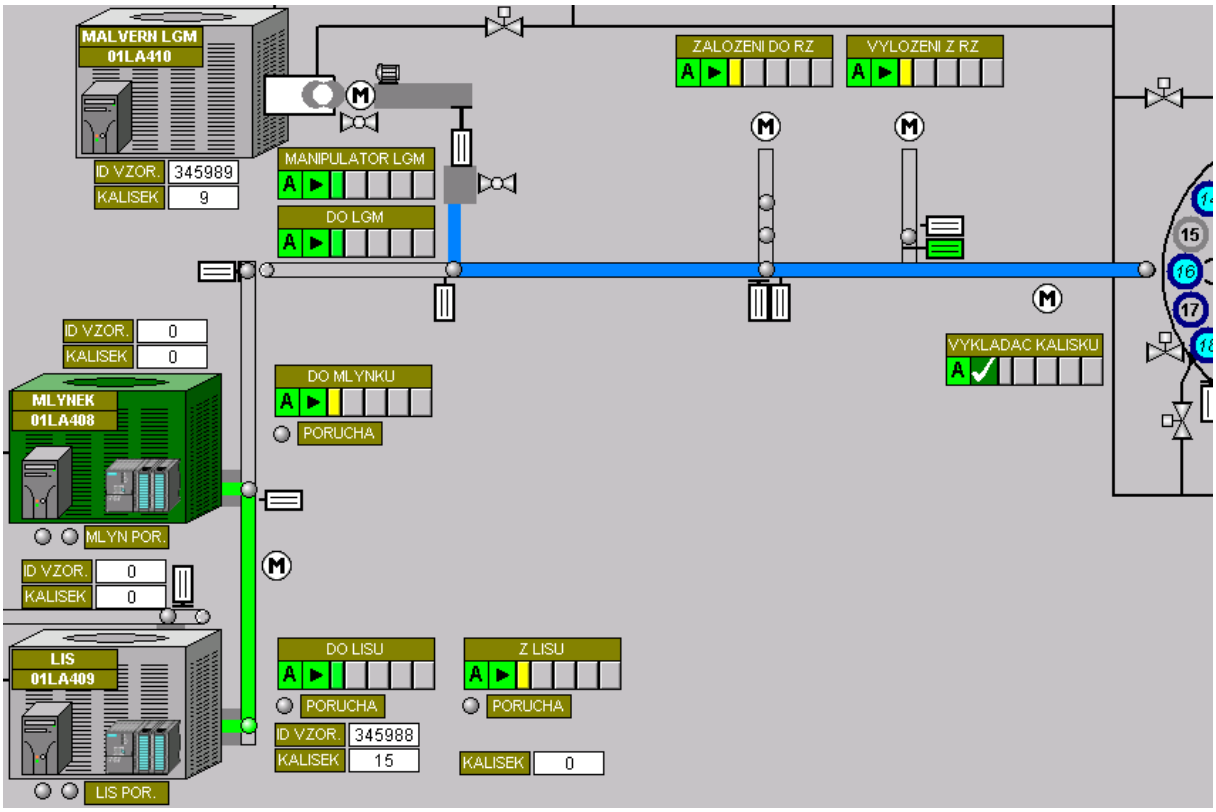
Detailnejšie zobrazenie objektov samotného laboratória je na obrazovke 01LA (príloha F). Na tejto obrazovke je kompletná vizualizácia všetkých objektov vygenerovaných z logiky (ventily, pohony, klapky) patriace do tejto hierarchie, čiže všetky zariadenia od prijímacej stanice až po röntgeny. Aby bolo zobrazenie čo najnázornejšie, vytvorili sme pre väčšinu objektov nové ikony ako piesty, zarážky a nasunovače. V starej vizualizácii boli všetky tieto objekty vizualizované ako ventily, z ktorých nebolo jasné, čo vlastne otvorenie tohto ventilu znamená v reálnej technológii. Toto bol problém hlavne v prijímacej stanici, kde je týchto ventilov veľa a sú uzavreté vo vnútri, takže ich nie je vidieť. Vďaka tomu je z vizualizácie na prvý pohľad jasné, čo sa v tejto časti deje, takže pri prípadnej poruche operátor hneď vidí aktuálnu situáciu.

Ďalej sme vytvorili novú ikonu pre sekvenciu, na originálnej totiž nie je nijak zobrazovaný aktuálny mód (Auto/Manual) ani stav sekvencie (Running, Hold, Aborted...). Nová ikona (obr. 4.20) tieto stavy zobrazuje, navyše sme na túto ikonu pridali políčko, ktorého farba sa dá riadiť zo sekvencie. Využívame ho v niektorých sekvenciách na odlíšenie stavu čakania (žltá farba) a práce so vzorkou (zelená).



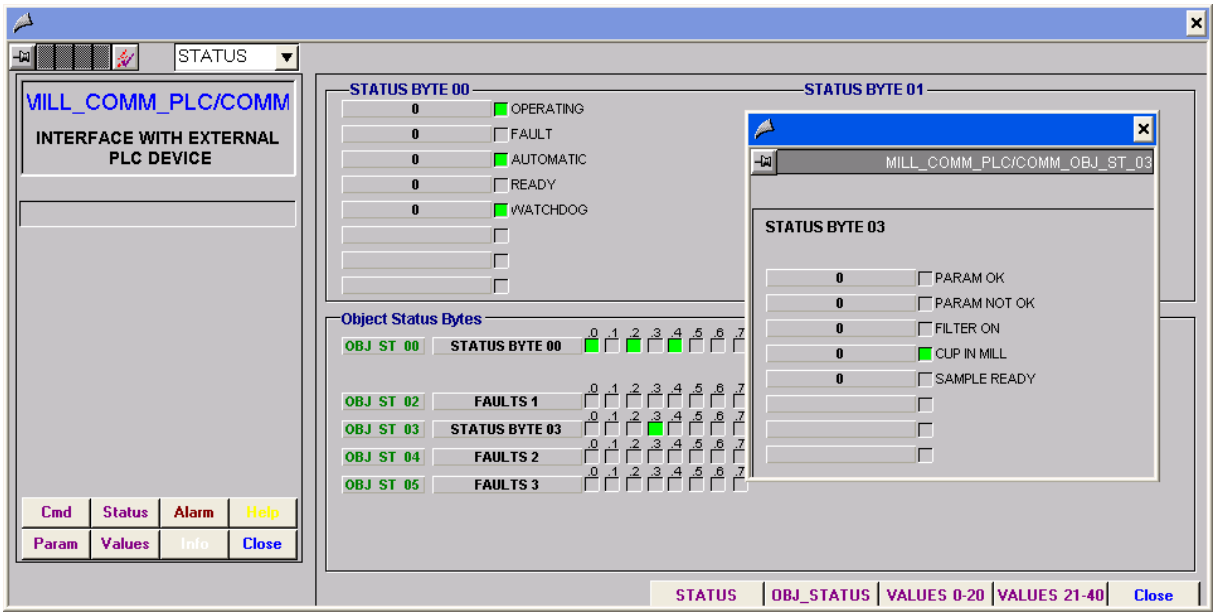
Obr. 4.20 - Nová ikona pre sekvenciu

Pri každej sekvencii sa nachádzajú dva políčka, v ktorých sa zobrazuje ID vzorky a číslo kalíšku, ktorý táto sekvencia aktuálne používa. Pri ceste kalíšku na spracovanie sa jeho cesta vyznačí zelenou čiarou (obr. 4.21), pri ceste prázdneho kalíšku späť je farba modrá. Zariadenia ako mlyn a lis sú tiež dodatočne dynamizované podľa stavov, ktoré tieto zariadenia posielajú nadradenému riadiacemu systému.

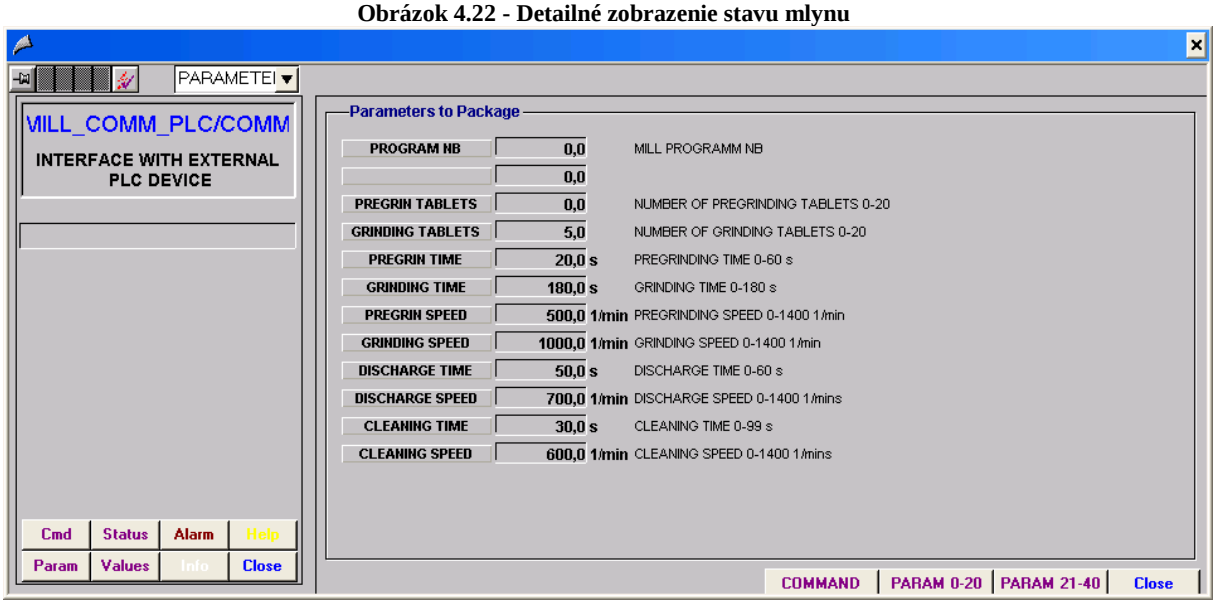


Obr. 4.21 - Zobrazenie ciest kalíškov

Detailnejší stav externých zariadení (lis, LGM, mlyn ...) je možné zistiť z detailných obrazoviek vďaka bloku U_EXT_DEVICE a U_EXT_DEVICE_DB. Aj keď sú tieto bloky rovnaké pre všetky zariadenia, zobrazujú sa v detailných obrazovkách popisy k jednotlivým zariadeniam podľa nakonfigurovaných parametrov týchto blokov v logike (CFC). Príklad zobrazenia bitových informácií o stave mlynu z komunikácie je na obr. 4.22.



Obrázok 4.22 - Detailné zobrazenie stavu mlynu



Obrázok 4.23 - Detailné zobrazenie parametrov posielaných pre mlyn

Na obr. 4.23 je detail okna s číselnými parametrami ktoré sa posielajú do mlynu, tieto parametre sú získané z databázy a pomocou bloku U_EXT_DEVICE posielané do mlynu. Na ďalších obrazovkách sú detailné vizualizácie samotných vzorkovačiek, ako príklad je v prílohe G obrazovka vzorkovačky 02VZ - MZ Silá.

5. Oživenie a testovanie laboratória

Nový riadiaci systém laboratória sme začali uvádzať do prevádzky v januári 2010 počas odstávky celej cementárne. Pretože išlo o prerobenie už existujúceho systému, nebolo potrebné robiť takmer žiadne zmeny v HW konfigurácii, všetky I/O periférie zostali pripojené tak ako v starom systéme. Napriek tomu sme pre kontrolu spravili testy všetkých binárnych signálov, pretože niektoré sa časom odstránili, prípadne nám nebola jasná ich funkcia.

Ďalej sme otestovali všetky akčné prvky v lokálnom režime tlačidlami z miesta, aby sme otestovali ich správne naviazanie v novom systéme. Je dôležité mať otestovanú funkčnosť všetkých prvkov skôr, ako sa spustia jednotlivé sekvencie, pretože nesprávnou funkciou niektorého prvku by mohlo dôjsť k poškodeniu zariadení. Po otestovaní v lokálnom režime sme opäť všetky objekty vyskúšali v režime Single, čo je režim umožňujúci ovládanie jednotlivých prvkov samostatne z vizualizácie.

Následne sme otestovali blok KARUSEL_STATUS a sekvenciu KARUSEL_POHYB spolu s manuálnym režimom karuselu a simulovanými operáciami s kalíškami. Počas tejto doby sme ešte po dohode s operátormi mierne upravili farebnú schému označenia kalíškov a nádob. Pri testovaní pohybu karuselu sme objavili chybu, kedy pri presune o jednu pozíciu sekvencia nestihla zachytiť správnu pozíciu a karusel sa najprv otočil o 360 stupňov a až potom zastal na správnej pozícii. Vyriešili sme to znížením oneskorenia pri detekcii signálu z čidla správnej polohy. Tak isto stavy kalíškov sa niekedy neaktualizovali správne, chyba bola v nesprávnom počte cyklov pri aktualizácii, kedy sa nepriamym adresovaním zapisovali dáta mimo správnu oblasť.

Aby sme mohli postupovať ďalej, museli sme nakonfigurovať komunikáciu s mlynom a lisom, a správne nastaviť parametre pre komunikačné bloky U_EXT_DEVICE. V tejto časti sme objavili určité nedostatky programu pre riadenie mlynu a lisu v ich riadiacich PLC. Pravdepodobne pre zjednodušenie starého systému tieto PLC nečakali na signál o pripravenosti odsávania, pretože program si tieto signály sám interne nastavoval a prepisoval. Bohužiaľ úpravám programu v týchto automatoch sme sa potrebovali vyhnúť, takže sme s touto vlastnosťou počítali a ošetrili ju v našom riadiacom systéme. Nakoniec sme mlyn aj lis nastavili správne do stavu v ktorom prijímal naše povely a preberal si správne parametre.

Vďaka tomu sme mohli postupne pokračovať v pripájaní a skúšaní jednotlivých sekvencií od tých najzákladnejších ako čistenie, vykladanie a zakladanie kalíšku, až po hlavné riadiace sekvencie pohybu kalíškov, prijímacej stanice a vzorkovacích staníc. Po dokončení tohto testovania sme teda mali plne funkčnú najnižšiu vrstvu riadenia laboratória. Následne na to sme spojili PLC s OPC Serverom a databázou a postupne otestovali komunikáciu PLC so službami pre röntgeny a LGM, pri čom sme otestovali funkčnosť blokov z vyššej vrstvy.

Tým sme dosiahli kompletné prepojenie aplikácie a databázy s riadiacim systémom v PLC a mohli sme otestovať kompletne celý priebeh od navzorkovania materiálu až po jeho analýzu. Celý tento postup trval dva týždne. Samozrejme sa v systéme prejavovalo veľa nedostatkov a chýb, ktoré sa vždy odhalia až na reálnej technológii. Jedným z hlavných problémov je oneskorenie pri komunikácii, ktoré spôsobuje OPC Server. Museli sme teda zabezpečiť to, aby sa pri komunikácii vždy najprv nastavili správne hodnoty a až po nastavenej dobe sa odoslal povel na zmenu týchto hodnôt.

Dosť veľký problém, ktorý sme hľadali niekoľko dní, spôsoboval zasekávanie sekvencií pohybu kalíškov. Tento problém bol spôsobený systémom fungovania sekvencií v SFC. Každý krok má totiž akcie Initialization, Processing a Termination. Pri vzájomnej blokácii sekvencií sa totiž pri splnení podmienky uvoľnenia sekvencia odblokuje a nastaví blokácie pre ostatné. Problém je ale v tom, že akcie Termination pôvodného kroku a Initialization toho nového sa vykonávajú až v ďalšom cykle PLC po detekcii splnenia podmienky. To spôsobovalo, že sa odblokovali dve sekvencie naraz, a navzájom sa dostali do deadlocku. Tieto hazardy sme postupne vyriešili u všetkých sekvencií kde mohli nastať.

Po vyriešení týchto problémov bolo laboratórium spustené v testovacej prevádzke a pravidelne zaťažované aby sa prejavili ďalšie prípadné problémy. Počas tejto doby sme ešte malými úpravami vylepšili rýchlosť postupu kalíškov, takže aj pri zapnutí všetkých vzorkovačiek je systém stále plynulo priechodný.

Posledný problém, ktorý sa doteraz nepodarilo uspokojivo vyriešiť je časté zlyhávanie LGM pri meraní, za túto chybu však nie je zodpovedný náš riadiaci systém. Túto chybu pravdepodobne spôsobuje ovládací program lasergranulometru, prípadne nejaká HW chyba.

6. Záver

Táto práca popisuje tvorbu návrhu a realizácie častí riadiaceho systému pre cementárne postaveného na systéme CEMAT od firmy Siemens. Čitateľovi poskytne základný prehľad vlastností a výhod tohto systému, ako aj základný technologický postup výroby cementu. Hlavný dôraz v práci je kladený na špecifické riešenia použité v týchto častiach, ktoré rozoberá detailnejšie. Výsledkom praktickej časti tejto práce sú vytvorené samotné projekty pre riadenie a vizualizáciu daných častí technológie.

Práca najprv rozoberá tvorbu dvoch častí riadiaceho systému pre cementáreň v Astane, hlavnom meste Kazachstanu. Prvá časť Expedícia vytvorená v prvej etape bola odskúšaná zákazníkom na FAT (Factory acceptance test) a na jeho základe prijatá vo februári 2009. Druhá časť, Drvenie a transport suroviny, bola súčasťou druhej etapy projektu. FAT tejto etapy prebehol v septembri 2009, projekt bol zákazníkom prijatý s výhradami, ktoré boli následne opravené a po overení akceptované. Termín nasadenia systému a jeho oživenia zatiaľ nebol stanovený, projekt je však na to pripravený.

Druhá časť práce sa zaoberá návrhom nového riadiaceho systému pre automatizované laboratórium v Radotíne. Nový systém kompletne nahradil pôvodný, vyriešil jeho problémy a zvýšil rýchlosť práce so vzorkami. Veľkým prínosom je tiež jeho kompletné začlenenie do centralizovaného systému riadenia celej cementárne a teda jednotnej vizualizácie. Vytvorený systém bol nasadený a oživený v období január - marec 2010, kedy úspešne prešiel testovacou prevádzkou a bol plne prevzatý zákazníkom. Systém v súčasnosti funguje v ostrej prevádzke dva mesiace bez akýchkoľvek závažnejších problémov.

Zoznam použitej literatúry

[1] Siemens Automation [online]

URL: http://www.automation.siemens.com

[2] Siemens: SIMATIC PCS 7 Brochure March 2007

[3] JIRÁSEK, J., Vavro, M.: *Anorganická pojiva* [online]

URL: http://geologie.vsb.cz/loziska/suroviny/anorganicka_pojiva.html

[4] Wikipedia: *Daylight saving time* [online]

URL: http://en.wikipedia.org/wiki/Daylight_saving_time

Zoznam použitých skratiek

SCL	- Structured Control Language
	- Štruktúrovaný riadiaci jazyk
POV	- Process object view
	- Zobrazenie procesných objektov
CFC	- Continuous function chart
	- Kontinuálny funkčný diagram
SFC	- Sequential function chart
	- Sekvenčný funkčný diagram
AS	- Automatization station
	- Automatizačná stanica
OS	- Operator station
	- Operátorská stanica
FC	- Function
	- Funkcia
FB	- Function block
	- Funkčný blok
DB	- Data block
	- Dátový blok
OB	- Organization block
	- Organizačný blok
FAT	- Factory acceptance test
	- Podnikový schvalovací test

Prílohy

Príloha A - Technologická obrazovka 0305_Limestone_Clay

Príloha B - Technologická obrazovka 07_Preblending

Príloha C - Technologická obrazovka 08_Raw_Transport

Príloha D - Technologická obrazovka Crushing - prehľadová

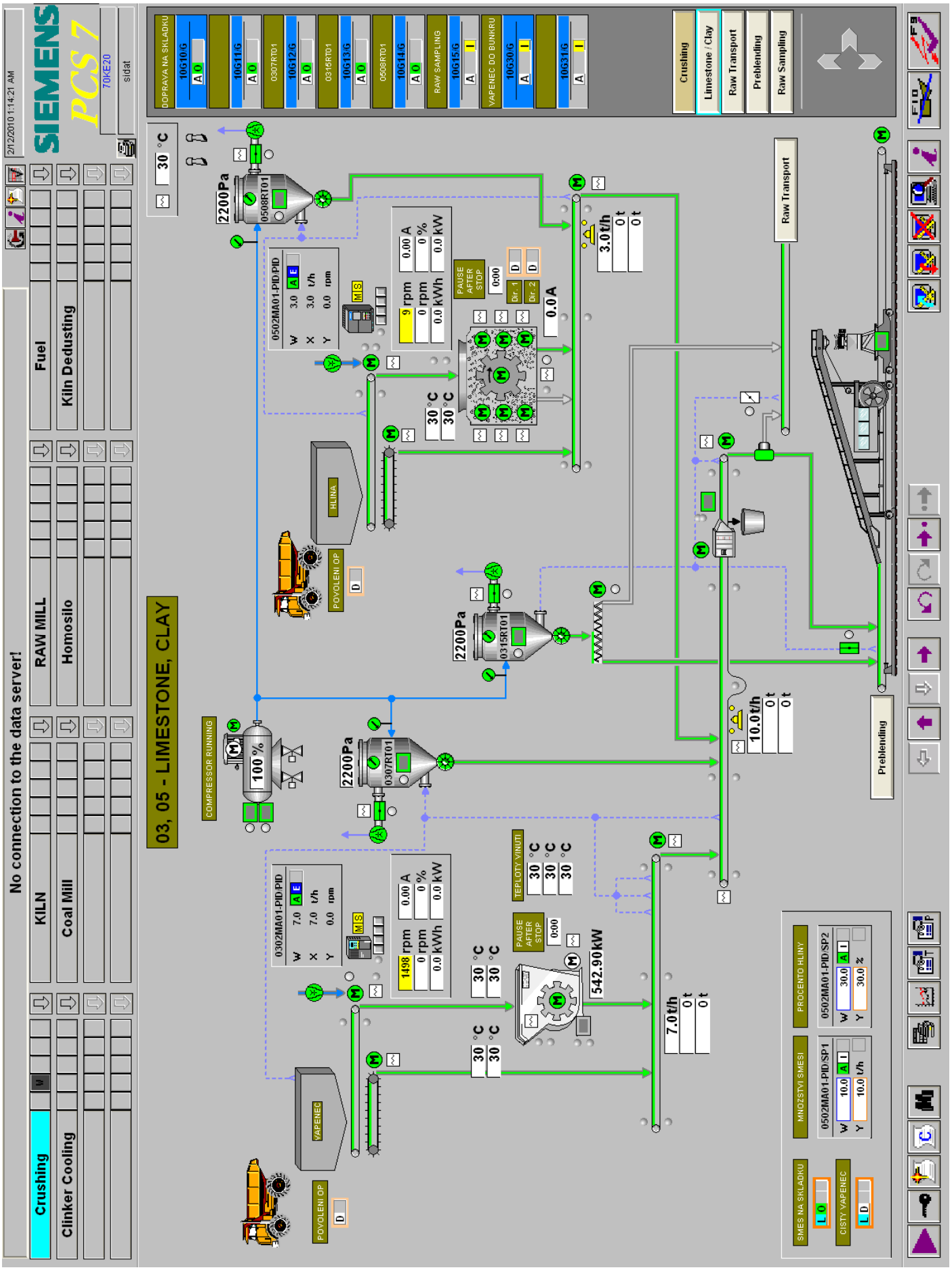
Príloha E - Technologická obrazovka Laboratoř - prehľadová

Príloha F - Technologická obrazovka 01LA - detail laboratória

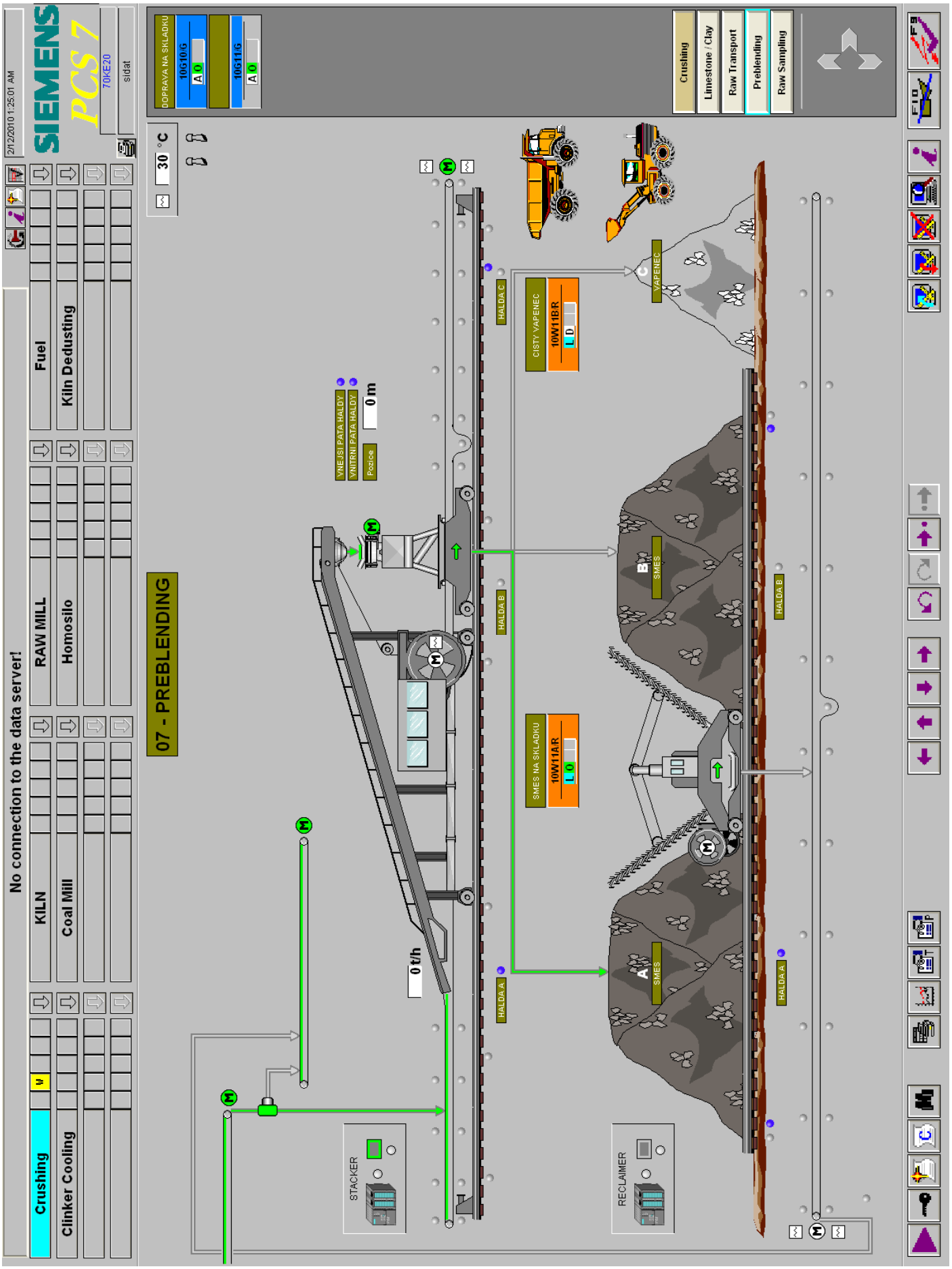
Príloha G - Technologická obrazovka 02VZ - detail vzorkovačky 2

Príloha H - Fotografie laboratória

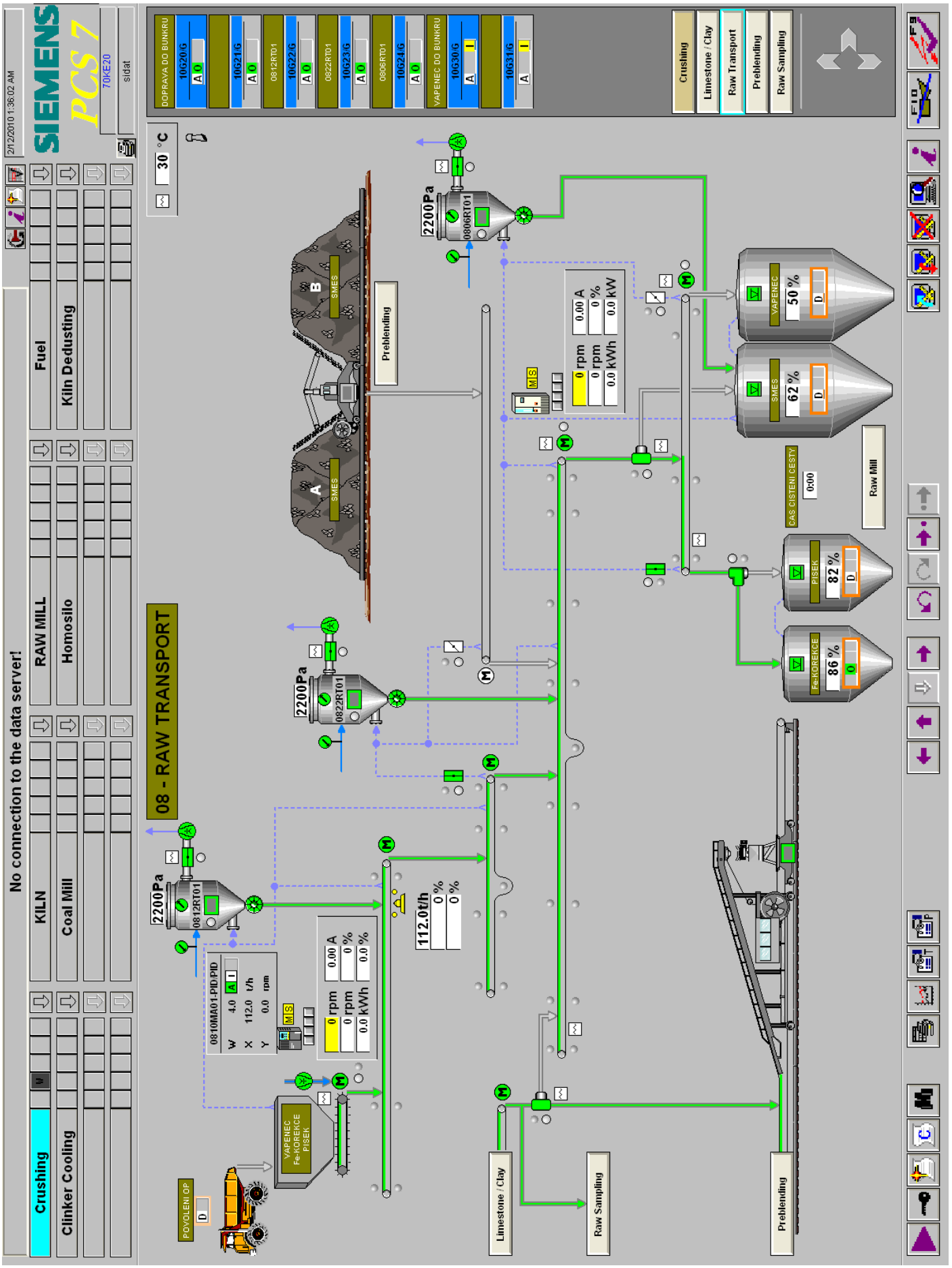
Príloha A - Technologická obrazovka 0305_Limestone_Clay



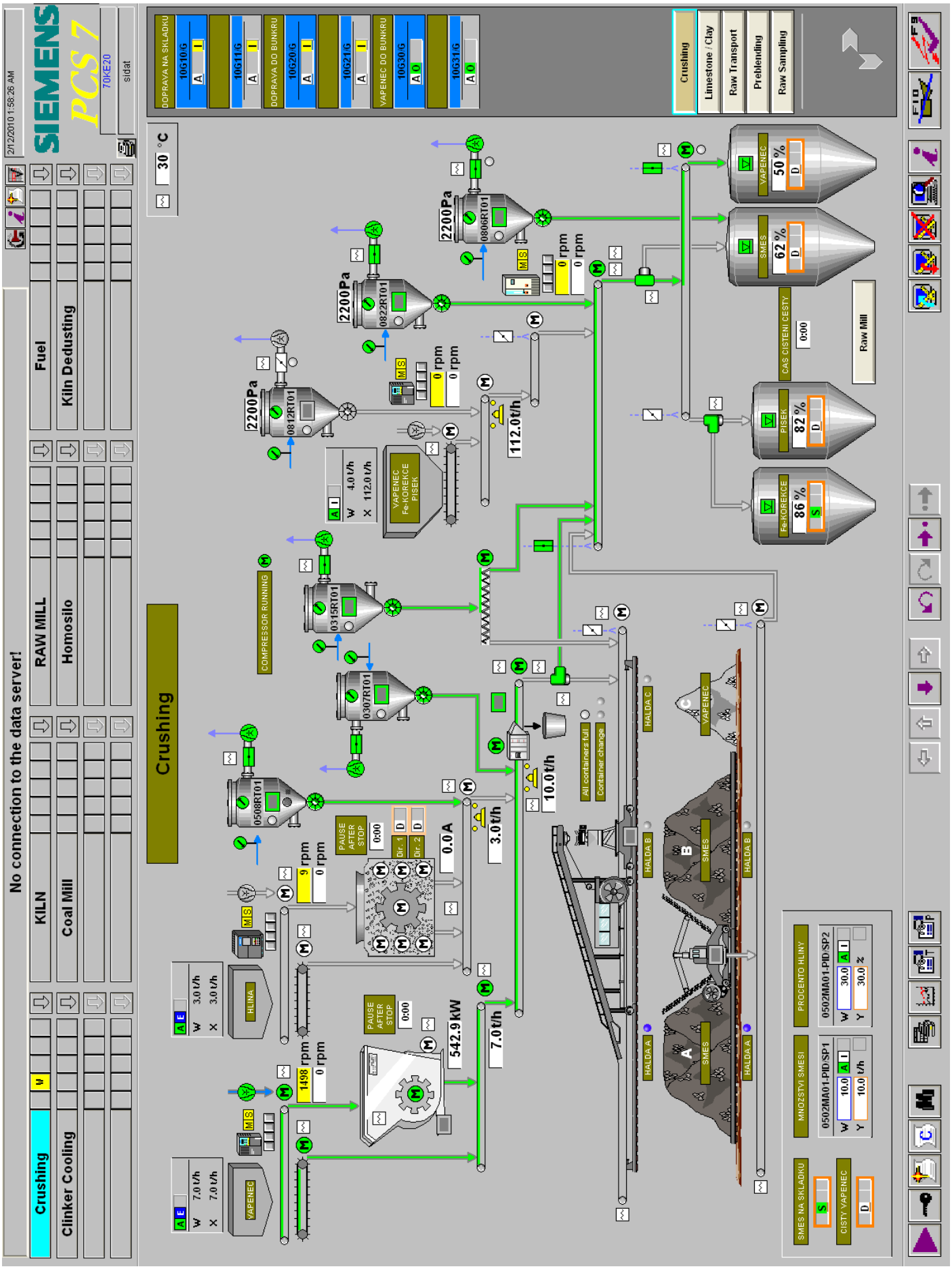
Príloha B - Technologická obrazovka 07_Preblending



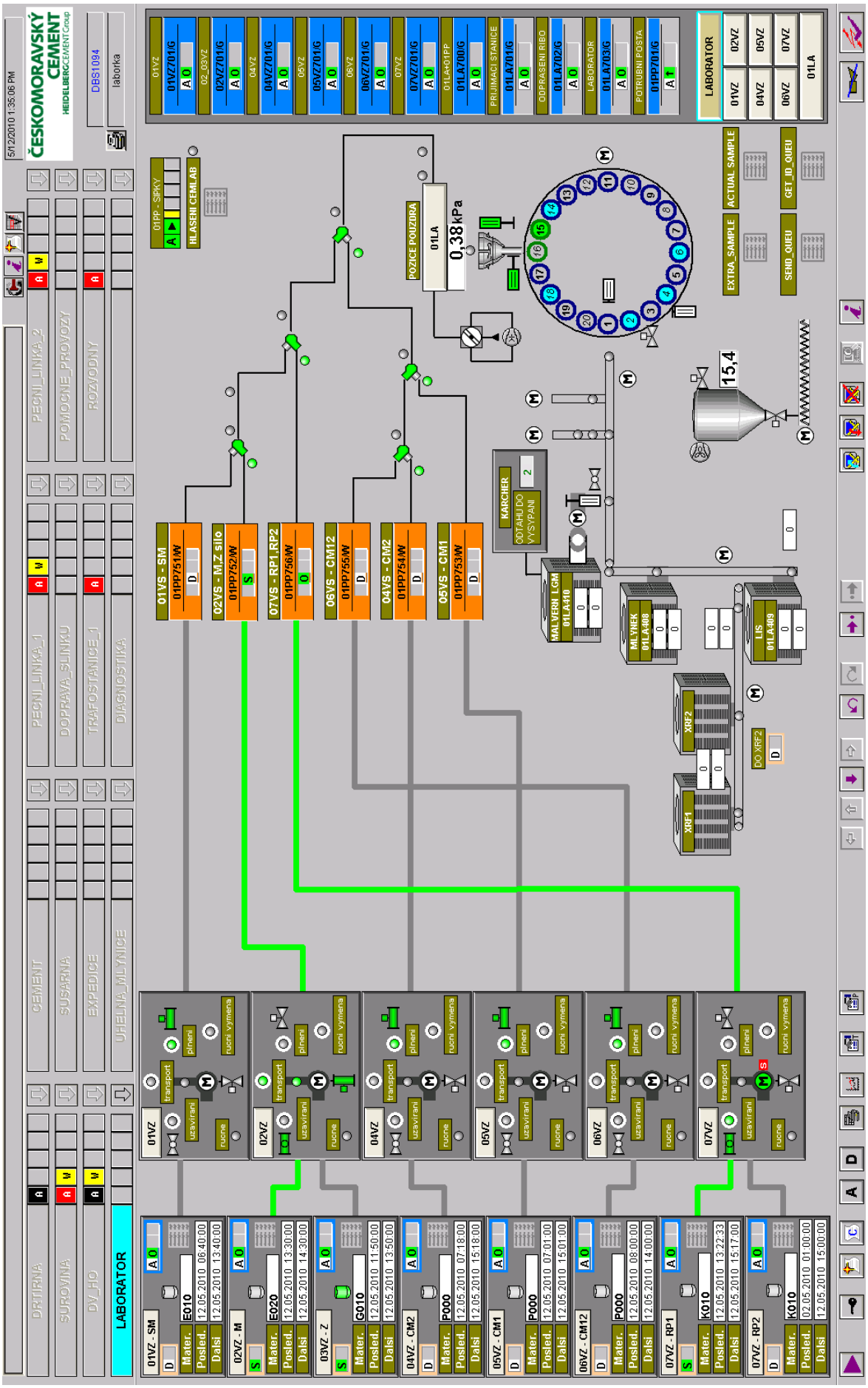
Príloha C - Technologická obrazovka 08_Raw_Transport



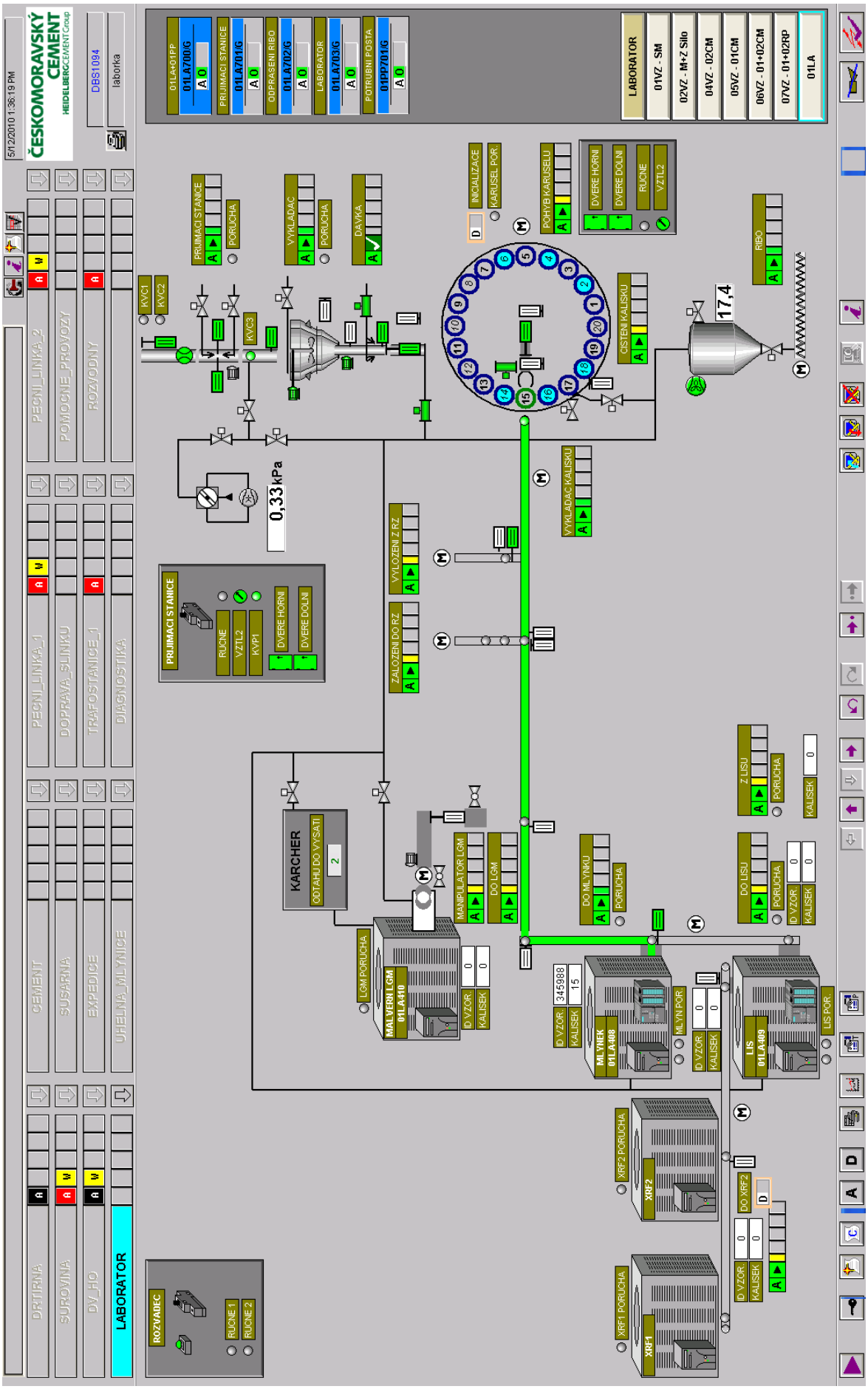
Príloha D - Technologická obrazovka Crushing - prehľadová



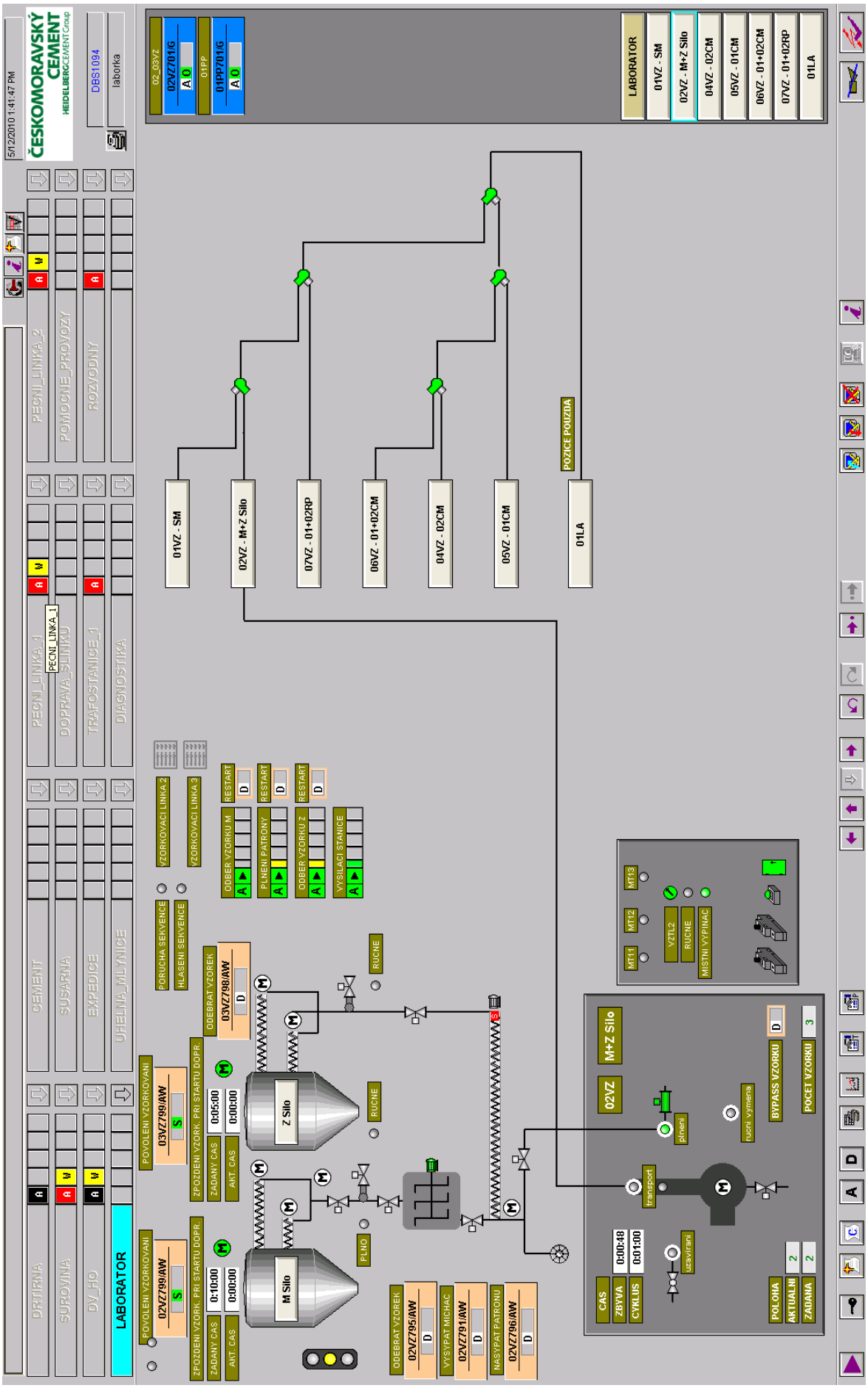
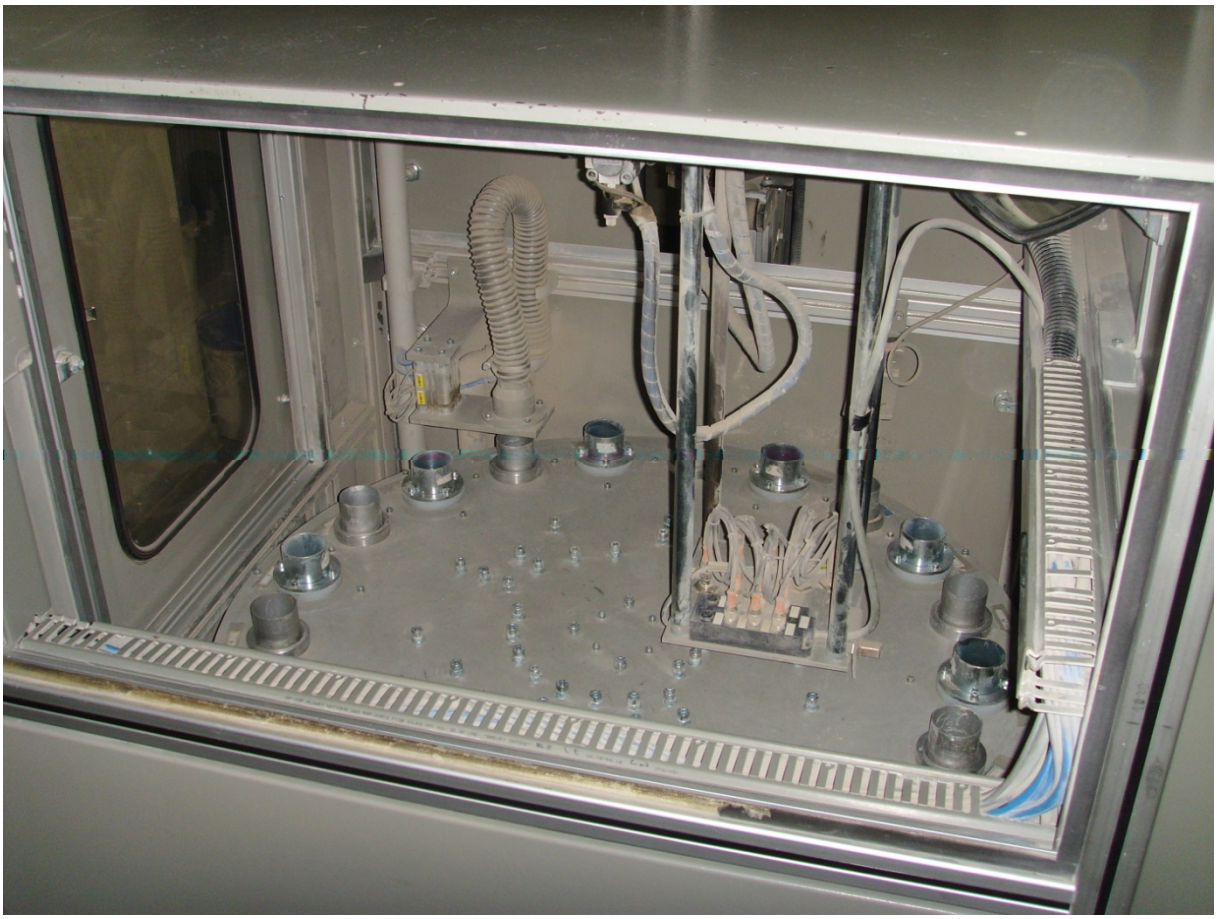
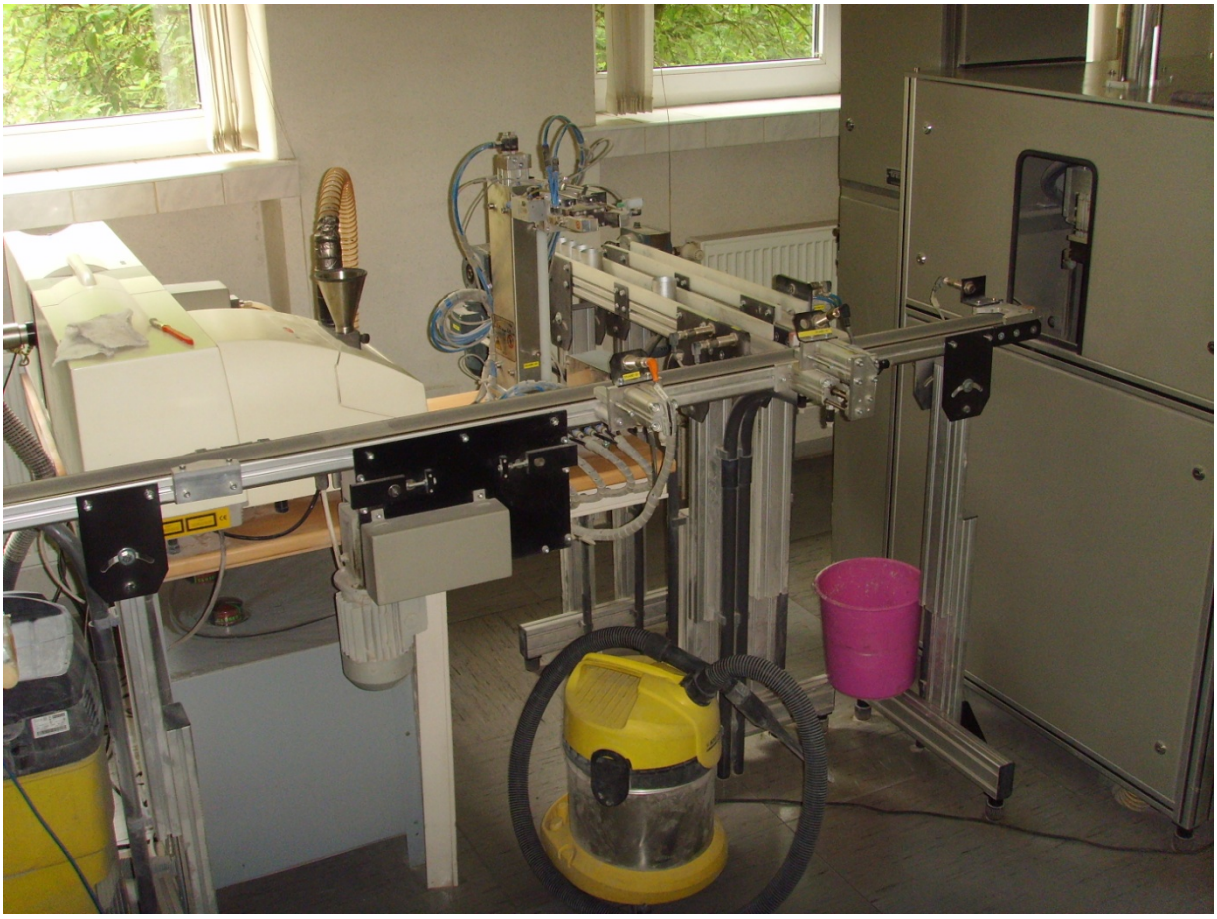
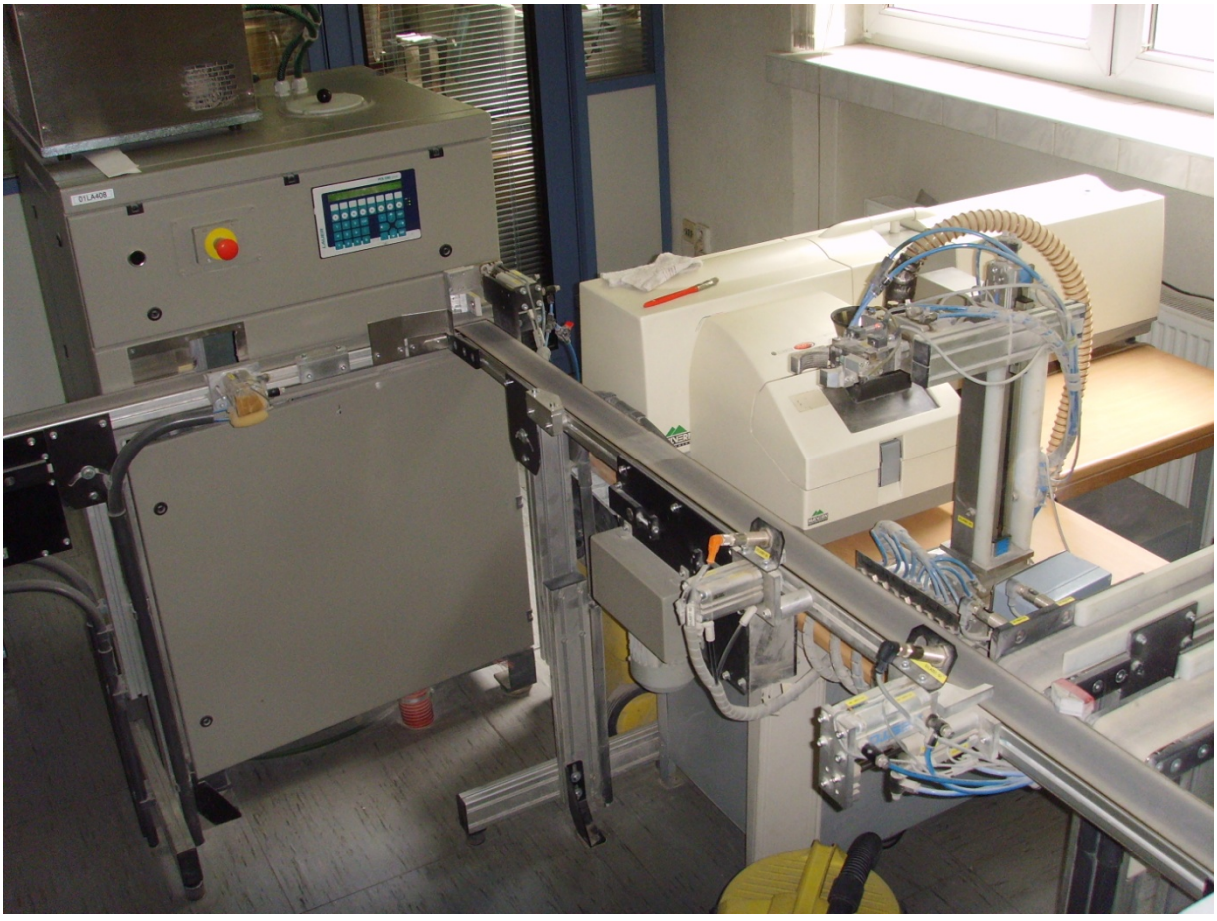
Príloha E - Technologická obrazovka Laboratoř - prehľadová



Príloha F - Technologická obrazovka 01LA - detail laboratória



Príloha G - Technologická obrazovka 02VZ - detail vzorkovačky 2



Príloha H - Fotografie laboratória