Online ISSN 1848-3380, Print ISSN 0005-1144
ATKAFF 54(1), 126–138(2013)

Franc Hanžič, Karel Jezernik, Slavko Cehner

Mechatronic Control System on a Finite-State Machine

DOI 10.7305/automatika.54-1.298
UDK 681.513.2-53; 004.41.021
IFAC 3.2; 4.7.6

Original scientific paper

This paper describes design using state-transition methodology. This state-transition methodology is straightforward, with a simply-perceived relation between the programming and the corresponding sequential function. The current operational function of the system is described as the current state of the system using state-transition programming. The state transition diagram or table describes the current state and the conditions for transition. The operation is transferred to a corresponding destination state when a set of conditions become valid for leaving the current state. Thus, the sequential operation is explicit, and any continuous conditions scanning (from command source and sensors) only include those that are pertinent for leaving the current state. The methodology is highly-structured and efficient, the programming tasks are readily comprehensible, and fault diagnostics can be easily included within the program's structure. The presented application of an automatic sliding-door illustrates the feasibility of this approach. This paper presents the MFSM (Modular Finite-State Machine), the ECA (Event-Condition-Action) system, motion generation, motion control with load estimation, and an example of a DSP (Digital Signal Processor) system. The limitations and attributes of each technique are discussed, and a state-table format is presented with the capability of representing parallel asynchronous sequential processes.

Key words: FSM, Software design, Adaptive automatic door motion, State flow, DSP

Mehatronički sustav upravljanja na automatu s konačnim brojem stanja. U ovom radu opisan je dizajn koristeći metodu prijelaza stanja. Metoda prijelaza stanja je izravna metoda s jednostavnim odnosom između programiranja i odgovarajuće funkcije. Trenutna funkcija sustava opisana je kao trenutno stanje sustava koristeći programiranje s prijelazom stanja. Dijagram ili tablica prijelaza stanja opisuje trenutno stanje i uvjete prijelaza u drugo stanje. Ako su zadovoljeni uvjeti prijelaza sustav prelazi u odgovarajuće stanje. Dakle, slijed je definiran eksplicitno te provjera kontinuiranih stanja sustava (upravljački signal ili senzori) obuhvaća samo ona stanja koja su značajna za prijelaz u sljedeće stanje. Metodologija je dobro strukturirana i učinkovita, programski zadaci su lako razumljivi i prepoznavanje pogreške može se lako uključiti u strukturu programa. Izvodljivost ovog pristupa ilustrirana je na primjeru automatskih kliznih vrata. Ovaj rad prikazuje modularni automat s konačnim brojem stanja, događajno uvjetovanu radnju, gibanje, upravljanje gibanjem s estimacijom tereta i primjer sustava s digitalnim signalnim procesorom. Komentirana su ograničenja i značajke svake od ovih metoda i prikazana je tablica stanja s mogućnošću prikazivanja paralelnih asinkronih slijednih procesa.

Ključne riječi: automat s konačnim brojem stanja, dizajn softvera, adaptivno gibanje automatskih vrata, digitalni signalni procesor

1 INTRODUCTION

Hybrid systems [1], covering the heterogeneous continuous, and discrete event natures of dynamic systems are well-known academic approaches. They are characterized by the interactions between those continuous parts governed by differential or difference equations, and the discrete-event part, traditionally described by finite-state machines [2-3], if then - else rules, temporal logic and discrete components, on-off switches, digital circuitry, software code, etc [4]. Hybrid systems switch between several operating modes, where each mode is governed by its own characteristic dynamic laws. Mode transitions are triggered by variable crossing-specific thresholds – "state events", by the elapses of certain time periods – "time events", or by external inputs – "input events" [5]. The typical hybrid systems are embedded systems, composed of dynamic components governed by logical/discrete decision components [6]. Hybrid systems require interdisciplinary approaches that exploit formal methods in computer science, control theory, and operational research. The motion method theory is used for jerk reduction [7-9]. Traditional control engineering practice favors recursive methods for describing motion structures. Recursive structures are much simpler to design, analyze, evaluate, and implement on microcontroller-based control platforms. A promising recursive description for event-driven systems [10-11] has been introduced recently. This paper applies this recently-developed recursive description of event-driven systems within the field of mechatronic systems [12], like an automatic sliding door.

The basic theory of FSM is simple with logical interactions. Such a design is preferred by companies with program design backgrounds. The program code must be logically written in program code and explained in papers because of the frequent changes of people working on projects. The disadvantage of such a design is the significant effort required to maintain organized program code on huge complex systems. The culprits for such a disadvantage are transitions, which can become intertwined within undesirable designs. Petri nets are mathematical modeling language for use on huge complex systems where the basic theory is more demanding than by FSM. The work bases of the FSM design improvements are to maintain organized program codes on complex systems without significant effort. The research is divided into modular FSM, and FSM with state sub-levels. The first technique describes a FSM within a modular method. The point of such a design is to divide huge processes into multiple smaller ones as asynchronous FSM processes. The connection between FSM modules is only implemented using FIFO queue event channels and not with transitions. The second technique describes FSM with multiple state sub-levels. Such a technique is recommended in systems with small complex systems for example a motion generator with seven basic states and multiple state sub-levels. The point of such a design is first to present the simplest basic system function, and then gradually expand the system's functionality in detail.

The essence of this work was to demonstrate a graphical software design method using FSM with state sub-levels. The point of FSM usage is to lower difficulties within software programming, debugging, and maintenance. FSM, using multiple state sub-level designs, parses the complex working process into small modular working processes. Such a program-code can be easier for debugging and maintenance. The contribution of this work is to demonstrate such a technique with multiple sub-states, on a custom-made mechatronic application with DSP controller. Such a design was proposed for the rapid software improvement of an automatic sliding-door control. Such a design technique is recommended for the rapid control development of mechatronic systems for robust control, without serious program bugs.

An S-shaped velocity motion generator designed in FSM with multiple state sub-levels improved the door's functionality (lower motor current peak, smoother movement, improved safety, decreased motor heating, etc.). Rapid prototyping using the DSP system for the door mechanism was included during the application. The experimental system was designed for the rapid prototyping of control algorithms for automatic door improvements.

2 SYSTEM DESCRIPTION

This is a system (Fig. 1) where the control system controls the application of automatic doors. The control system receives certain information (inputs) from the application, and generates actions (outputs) that affect it. The control system is realized using simple logical conditions:

IF (input conditions) THEN outputs

The input conditions represent logical expressions formulated according to Boolean algebra:

door_open

door_closed AND person_detected

command_open OR person_detected

Where "AND" and "OR" are the logical operators. All input conditions have Boolean values (true, false). The outputs represent the control system's states (open, closed, locked etc.). Each previous state is stored within a variable. The new state depends on the previous state and the input conditions. A switching algorithm between states with input and state conditions is the core of the FSM. The software design can be implemented with the help of a transition matrix or state machine diagram.

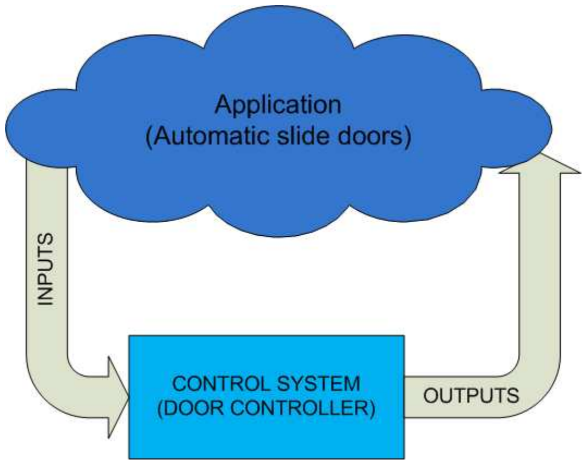


Fig. 1. An application using the control system

The door control software designed in FSM is divided into continuous and discrete-event spaces (Fig. 2). The continuous space contains a continuous data stream both from and to the sensors and actuators that are used on the automatic doors. The discrete event space is based on FSM with event interrupts.

The transition matrix (Fig. 3) is a table with represented states, transitions and events of the observed system. The event occurs when a specific input or function condition is met. The transition matrix can cover various forms, depending on the person who designed the FSM. Illegal transitions are represented by the red fields.

The state transition diagram (Fig. 4) is a graphical representation, which is described within the transition matrix (Fig. 3). It consists of circles (states) and connections between them (transitions).

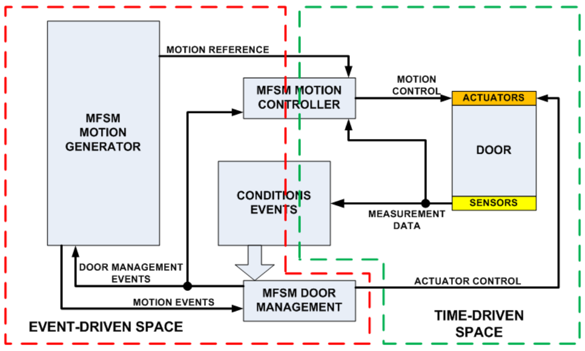
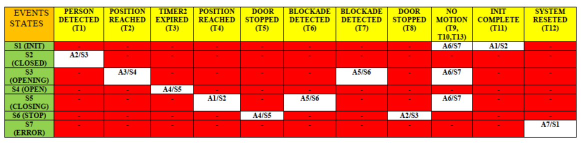


Fig. 2. Door control software



EVENTS STATES	PERSON DETECTED (T1)	POSITION REACHED (T2)	TIMER2 EXPIRED (T3)	POSITION REACHED (T4)	DOOR STOPPED (T5)	BLOCKADE DETECTED (T6)	BLOCKADE DETECTED (T7)	DOOR STOPPED (T8)	NO MOTION (T9, T10,T13)	INIT COMPLETE (T11)	SYSTEM RESETED (T12)
S1 (INIT)	-	-	-	-	-	-	-	-	A6/S7	A1/S2	-
S2 (CLOSED)	A2/S3	-	-	-	-	-	-	-	-	-	-
S3 (OPENING)	-	A3/S4	-	-	-	-	A5/S6	-	A6/S7	-	-
S4 (OPEN)	-	-	A4/S5	-	-	-	-	-	-	-	-
S5 (CLOSING)	-	-	-	A1/S2	-	A5/S6	-	-	A6/S7	-	-
S6 (STOP)	-	-	-	-	A4/S5	-	-	A2/S3		-	-
S7 (ERROR)	-	-	-	-	-	-	-	-	-	-	A7/S1

Fig. 3. Transition matrix

The door control system consists of a few different tasks. The microcontrollers don't support parallel task execution like FPGAs, so an additional operating system must be implemented within the microcontroller. The operating system RTOS (Real Time Operating System) for embedded applications is used for parallel task execution on the microcontrollers. RTOS can also be implemented within the FSM design. The door application uses the following tasks:

1. Input reading task (creates various events from the sensor's signal read);
2. Control task (door motion and speed control);
3. Door management task (safety functions, door management, hardware control, diagnostics, flash memory logging etc.)

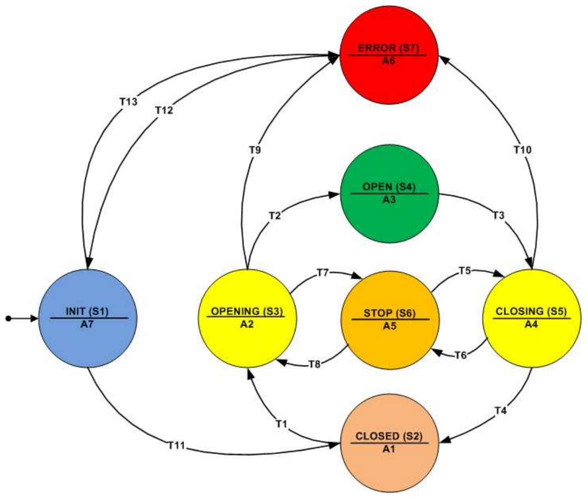


Fig. 4. State transition diagram

4. Communication task (CAN and Serial RS232)
5. Door operator task (generates events from a command switch or any other operator via CAN communication)

The improved motion control algorithm (Fig. 5) is used [13] within the door's system. Feed-back control consists of three reference inputs (acceleration, speed, and position) that are given from the motion generator and two actual measure inputs (position, speed) that are given from the incremental encoder. The feed-back control is tuneable with two parameters K_p and K_v). The K_v parameter affects the quality of the velocity control, and the K_p parameter affects the quality of the positional control. Both parameters affect the system's control speed performance. The feed-back output represents an acceleration reference that is multiplied by the estimated mechanism's moving mass ($\hat{m}$). The multiplication result is the estimated force of the mechanism. The feed-forward control eliminates the error of a poorly-specified estimated mechanism's moving mass, and any frictions in the gears, bearings, and guides. The parameter h serves as a low-pass filter that eliminates noise during speed measures, so this parameter has a small number, depending on the measure quality. The mechanism starts trembling if the parameter h is set too high, if set too low the mechanism's response-time increases. The parameter h is defined using practical experiments, by considering the optimal control response. The total reference force is the sum of the mechanism's estimated force and the force from the feed-forward control. The motor is based on rotation, so the motor references are torque based. Torque is transformed into force and vice-versa because the mechanism is linear-based with gears and a belt-pulley

(2). The transform calculation depends on the gear ratio i and pulley radius r. The motor torque constant K_m is given in the motor datasheet.

$$M = \frac{F \cdot r}{i} \tag{1}$$

The following explanation about FSM design is pointed out in the control task, where the door motion and motion control are included. An experiment (Fig. 6, and Fig. 7) is included on a DSP system with linear mechanism and a brushless AC motor. The brushless AC motors are commutated with sinusoidal current shapes for less torque ripple. The DSP system was designed for control algorithm rapid prototyping [15-16]. An improved motion control was integrated within the door motion FSM. The experimental environment was designed for the test purposes of the control algorithms. The PC is used for control designing, control response analysis, and data measures.

3 DOOR-MOTION

A trapezoidal speed profile is used in current doors. This kind of profile doesn't have a continuous movement between speed-crossings. The door movement's smoothness, increases motor heating, and more energy consumption is created by the trapezoidal profile. The event-based S speed profile was designed to negate such disadvantages. The idea of an event-based system was to design motion control within a graphical-based environment. The point of such research was to improve software debugging, execution tracking, monitoring, and documentation, and to provide explanations of the control algorithm for the next generation of software designers within companies. The basics of S speed-profile are known and widely-used in motion controllers around the world.

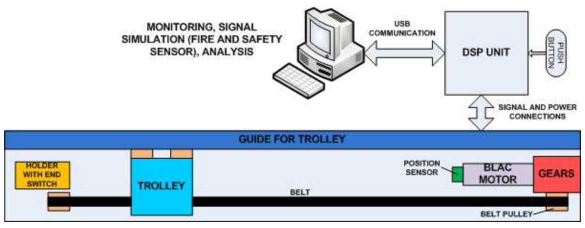


Fig. 6. Experiment's environment

This means that the algorithm core is known but the design has been changed for automatic door control purposes.

The door application consists of different multiple processes. One such process consists of a motion generator with the S based speed-profile. The S speed-profile (Fig. 8)

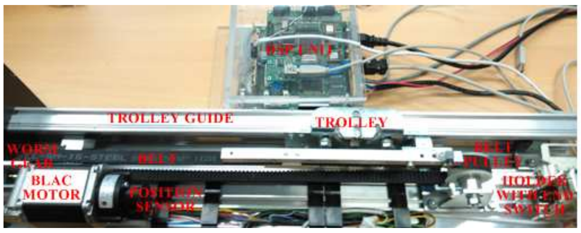


Fig. 7. Experimental test-bed

was developed based on MFSM for continuous movement between speed-crossings. MFSM development contributes software development with low based integration within a multi-process. The profile has to be adaptable to motion length, maximum jerk, acceleration, and speed. The jerk controls the smoothness level between the speed crossings. The smoothness level decreases as the jerk increases. The trapezoidal profile can be generated by a high-jerk. The maximum acceleration and velocity of the door must be controllable because of door-safety. The acceleration and speed must be lower for heavier doors than lighter ones. These rules are implemented because heavier doors are more dangerous and can lead to serious injuries for people during collisions. The motion length is initialized at the door start-up, and depends on the door type.

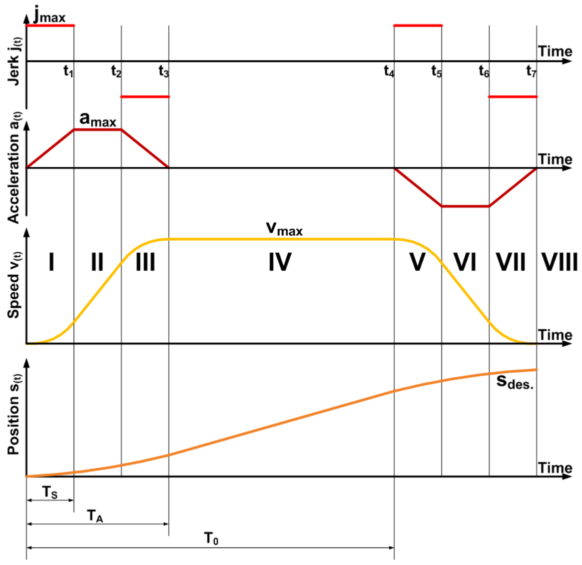


Fig. 8. S speed profile generation

Where is:

jmax – maximum jerk

amax – maximum acceleration

vmax – maximum velocity

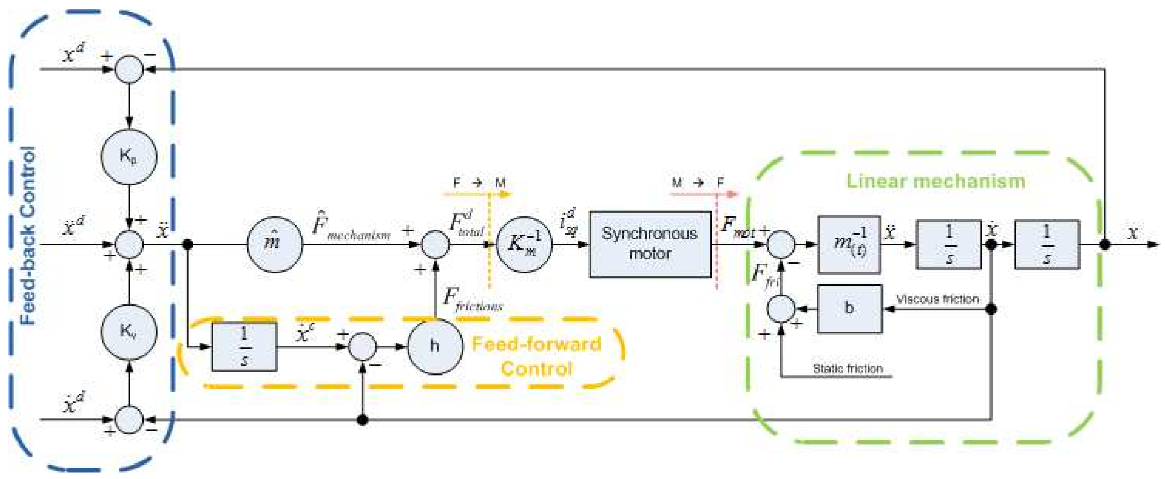


Fig. 5. Mechanism control algorithm

sdes – desired position

TS – jerk-time

TA – acceleration time

T0 – time to deceleration

The S speed profile is generated with the following equations for acceleration (2), velocity (3), and position (4). The equations represent the mathematical models of other authors [7-9]. These authors have presented similar objectives for developing a motion system for better motion properties. The crossing between the positive and negative speed profiles represents a rotational change by rotary motors and a directional change by linear-motors.

$$a_{(t)} = \begin{cases} j_{\max} \cdot t & 0 < t \le t_1 = T_S \\ a_{\max} & t_1 < t \le t_2 = T_A - T_S \\ -j_{\max} \cdot (t - t_2) + a_{\max} & t_2 < t \le t_3 = T_A \\ 0 & t_3 < t \le t_4 = T_0 \\ -j_{\max} \cdot (t - t_4) & t_4 < t \le t_5 = T_0 + T_S \\ -a_{\max} & t_5 < t \le t_6 = T_0 + T_A - T_S \\ j_{\max} \cdot (t - t_6) - a_{\max} & t_6 < t \le t_7 = T_0 + T_A \end{cases} \tag{2}$$

$$v_{(t)} = \begin{cases} \frac{1}{2} \cdot j_{\max} \cdot t^2 & 0 < t \le t_1 = T_S \\ a_{\max} \cdot (t - t_1) + v_{(t_1)} & t_1 < t \le t_2 = T_A - T_S \\ -\frac{1}{2} \cdot j_{\max} \cdot (t - t_2)^2 + a_{\max} \cdot (t - t_2) + v_{(t_2)} & t_2 < t \le t_3 = T_A \\ v_{(t_3)} & t_3 < t \le t_4 = T_0 \\ -\frac{1}{2} \cdot j_{\max} \cdot (t - t_4)^2 + v_{(t_4)} & t_4 < t \le t_5 = T_0 + T_S \\ -a_{\max} \cdot (t - t_5) + v_{(t_5)} & t_5 < t \le t_6 = T_0 + T_A - T_S \\ \frac{1}{2} \cdot j_{\max} \cdot (t - t_6)^2 - a_{\max} \cdot (t - t_6) + v_{(t_6)} & t_6 < t \le t_7 = T_0 + T_A \end{cases} \tag{3}$$

$$s_{(t)} = \begin{cases} \frac{1}{6} \cdot j_{\max} \cdot t^3 & 0 < t \le t_1 = T_S \\ \frac{1}{2} \cdot a_{\max} \cdot (t - t_1)^2 + v_{(t_1)} \cdot (t - t_1) + s_{(t_1)} & t_1 < t \le t_2 = T_A - T_S \\ -\frac{1}{6} \cdot j_{\max} \cdot (t - t_2)^3 + \frac{1}{2} \cdot a_{\max} \cdot (t - t_2)^2 + v_{(t_2)} \cdot (t - t_2) + s_{(t_2)} & t_2 < t \le t_3 = T_A \\ v_{(t_3)} \cdot (t - t_3) + s_{(t_3)} & t_3 < t \le t_4 = T_0 \\ -\frac{1}{6} \cdot j_{\max} \cdot (t - t_4)^3 + v_{(t_4)} \cdot (t - t_4) + s_{(t_4)} & t_4 < t \le t_5 = T_0 + T_S \\ -\frac{1}{2} \cdot a_{\max} \cdot (t - t_5)^2 + v_{(t_5)} \cdot (t - t_5) + s_{(t_5)} & t_5 < t \le t_6 = T_0 + T_A - T_S \\ \frac{1}{6} \cdot j_{\max} \cdot (t - t_6)^3 - \frac{1}{2} \cdot a_{\max} \cdot (t - t_6)^2 + v_{(t_6)} \cdot (t - t_6) + s_{(t_6)} & t_6 < t \le t_7 = T_0 + T_A \end{cases} \tag{4}$$

where:

$$\begin{aligned} j_{\max} &- \text{maximum jerk} \\ a_{\max} &- \text{maximum acceleration} \\ v_{(t_n)} &- \text{velocity at end of time (n)} \\ s_{(t_n)} &- \text{position at end of time (n)} \end{aligned}$$

The time intervals (TS, TA, and T0) are calculated using different equations (Table 1, and Table 2) because of changeable maximum jerk, acceleration, velocity, and the desired position.

Velocity condition: $v < \frac{a_{\max}^2}{j_{\max}}$		
Position conditions:	$s_{des} \geq 2 \cdot \sqrt{\frac{v_{\max}^3}{j_{\max}}}$	$s_{des} < 2 \cdot \sqrt{\frac{v_{\max}^3}{j_{\max}}}$
T_S	$\sqrt{\frac{v_{\max}}{j_{\max}}}$	$\sqrt[3]{\frac{v_{\max}}{2 \cdot j_{\max}}}$
T_A	$2 \cdot T_S$	$2 \cdot T_S$
T_0	$\frac{v_{\max}}{s_{des}}$	T_A

Table 1. Time interval equation conditions 1/2

Velocity condition: $v \geq \frac{A_{\max}^2}{j_{\max}}$			
Position conditions:	$s_{des} \geq \frac{v_{\max}^2}{a_{\max}} + \frac{a_{\max} \cdot v_{\max}}{j_{\max}}$	$\frac{v_{\max}^2}{a_{\max}} + \frac{a_{\max} \cdot v_{\max}}{j_{\max}} > s_{des} \geq \frac{2 \cdot a_{\max}^3}{j_{\max}^2}$	$s_{des} < \frac{2 \cdot a_{\max}^3}{j_{\max}^2}$
T_S	$\frac{a_{\max}}{j_{\max}}$	$\frac{a_{\max}}{j_{\max}}$	$\sqrt[3]{\frac{v_{\max}}{2 \cdot j_{\max}}}$
T_A	$\frac{v_{\max}}{a_{\max}} + T_S$	$\frac{-a_{\max}^2 + \sqrt{a_{\max}^4 + 4 \cdot a_{\max} \cdot j_{\max}^2 \cdot s_{des}}}{2 \cdot j_{\max} \cdot a_{\max}} + T_S$	$2 \cdot T_S$
T_0	$\frac{s_{des}}{v_{\max}}$	T_A	T_A

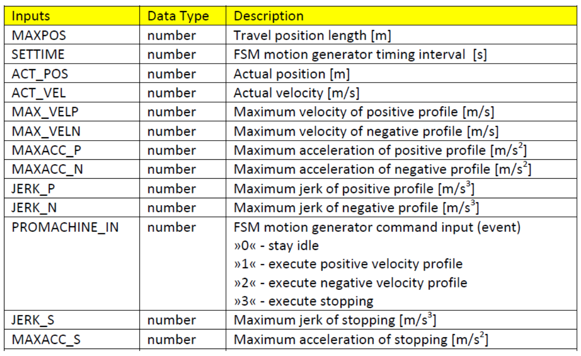
Table 2. Time interval equation conditions 2/2

4 MOTION BASED ON FSM

The FSM door motion was designed in Matlab/Simulink – StateFlow. The FSM represents a graphical block with inputs (Fig. 9) and outputs (Fig. 10).

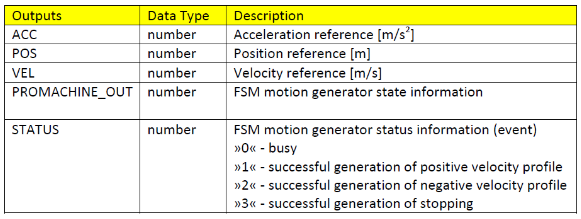
The FSM motion-generator has 14 different inputs. The input PROMACHINE_IN is connected to the door management FSM, which gives commands to the FSM motion-generator. All other inputs (ACT_POS and ACT_VEL are excluded) are connected to the door-system's initialization, which calculates all the parameters needed for door control, including the conditions in (Table 1, and Table 2).

The FSM motion-generator with 5 outputs represents 3 pieces of reference data (acceleration, velocity, and position). The reference data is connected to the control system. The output PROMACHINE_OUT is intended for the



Inputs	Data Type	Description
MAXPOS	number	Travel position length [m]
SETTIME	number	FSM motion generator timing interval [s]
ACT_POS	number	Actual position [m]
ACT_VEL	number	Actual velocity [m/s]
MAX_VELP	number	Maximum velocity of positive profile [m/s]
MAX_VELN	number	Maximum velocity of negative profile [m/s]
MAXACC_P	number	Maximum acceleration of positive profile [m/s^2]
MAXACC_N	number	Maximum acceleration of negative profile [m/s^2]
JERK_P	number	Maximum jerk of positive profile [m/s^3]
JERK_N	number	Maximum jerk of negative profile [m/s^3]
PROMACHINE_IN	number	FSM motion generator command input (event) »0« - stay idle »1« - execute positive velocity profile »2« - execute negative velocity profile »3« - execute stopping
JERK_S	number	Maximum jerk of stopping [m/s^3]
MAXACC_S	number	Maximum acceleration of stopping [m/s^2]

Fig. 9. FSM motion-generator inputs



Outputs	Data Type	Description
ACC	number	Acceleration reference [m/s^2]
POS	number	Position reference [m]
VEL	number	Velocity reference [m/s]
PROMACHINE_OUT	number	FSM motion generator state information
STATUS	number	FSM motion generator status information (event) »0« - busy »1« - successful generation of positive velocity profile »2« - successful generation of negative velocity profile »3« - successful generation of stopping

Fig. 10. FSM motion-generator outputs

FSM motion-generator diagnostic – states toggle information. The status output is connected to the door management FSM for the event transmission of the FSM motion-generator's status.

The development of FSM in sub-layers contributes to program functionality explanations regarding graphic presentations. This method explains the program's functionality as multiple-difficulty classes. The simplest processes cover only a few state sub-levels but the complex processes can cover several state sub-levels. The software development can best be explained as being like lego building bricks divided in difficulty levels. The project "newbie employee" can easily adapt to the existing work. The transition's knits are reduced and maintain program code traceability. Any possible code bugs can be more precisely oriented with state sub-layer identification. Such a technique is show in the state flow diagram of the motion generator. The main level represents the basic operation (Table 3, and Fig. 11). The advantage of such a technique is the independence of the program's languages.

The states (init., positive, negative, and stop) have two sub-levels. Let's take the state-positive for example. The first sub-level contains positive motion profile sector-switching (sectors I to VIII – see Fig. 8). The positive motion consists of 8 first sub-states (Table 4 ,and Fig. 12). The states at the first sub-level have two numbers (Sxy), where x is the state number of the main level and y the state number of first sub-level.

Events	Description
E1	Command event (positive or negative)
E2	Successful INIT with command event positive
E3	Successful INIT with command event negative
E4	Blockade event by opening
E5	Blockade event by closing or event of detected person
E6	Door has stopped event
E7	Door is open event
E8	Door is closed event
Actions	**Description**
A1	Execute INIT.
A2	Execute positive motion reference (execute sector I)
A3	Execute negative motion reference (execute sector I)
A4	Execute STOP
A0	Execute IDLE

Transition table								
Events / States	E1	E2	E3	E4	E5	E6	E7	E8
S0 (IDLE)	A1/S1	-	-	-	-	-	-	-
S1 (INIT.)	-	A2/S2	A3/S3	-	-	-	-	-
S2 (POSITIVE)	-	-	-	A4/S4	-	-	A0/S0	-
S3 (NEGATIVE)	-	-	-	-	A4/S4	-	-	A0/S0
S4 (STOP)	-	-	-	-	-	A0/S0	-	-

Table 3. FSM motion-generator transition matrix - main level

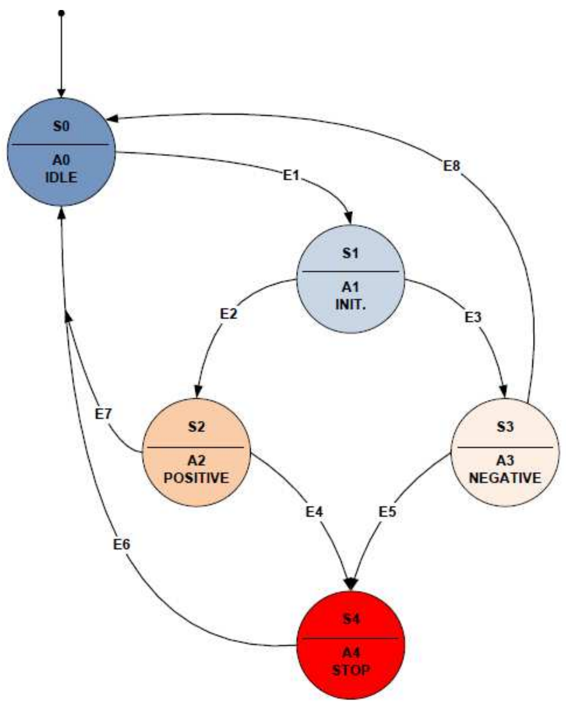


Fig. 11. FSM motion-generator state flow diagram - main level

The transition table's blue fields represent the events and actions from the main level. Each state in the sub-level is also conditioned from the main level.

The states at the first sub-level have states in the second sub-level. The second sub-level finally concludes the equations for motion calculations (1, 2, and 3). The state

Events	Description
E20	Time t_1 expired
E21	Time t_2 expired
E22	Time t_3 expired
E23	Time t_4 expired
E24	Time t_5 expired
E25	Time t_6 expired
E26	Time t_7 expired
Actions	**Description**
A20	Execute sector II
A21	Execute sector III
A22	Execute sector IV
A23	Execute sector V
A24	Execute sector VI
A25	Execute sector VII
A26	Execute sector VIII

	Transition table							
Events / States	E20	E21	E22	E23	E24	E25	E26	E4 / E6
S20 (I)	A_{20}/S_{21}	-	-	-	-	-	-	A_4/S_4
S21 (II)	-	A_{21}/S_{22}	-	-	-	-	-	A_4/S_4
S22 (III)	-	-	A_{22}/S_{23}	-	-	-	-	A_4/S_4
S23 (IV)	-	-	-	A_{23}/S_{24}	-	-	-	A_4/S_4
S24 (V)	-	-	-	-	A_{24}/S_{25}	-	-	A_4/S_4
S25 (VI)	-	-	-	-	-	A_{25}/S_{26}	-	A_4/S_4
S26 (VII)	-	-	-	-	-	-	A_{26}/S_{27}	A_4/S_4
S27 (VIII)	-	-	-	-	-	-	-	A_0/S_0 / A_4/S_4

Table 4. FSM motion-generator transition matrix – first sub-level

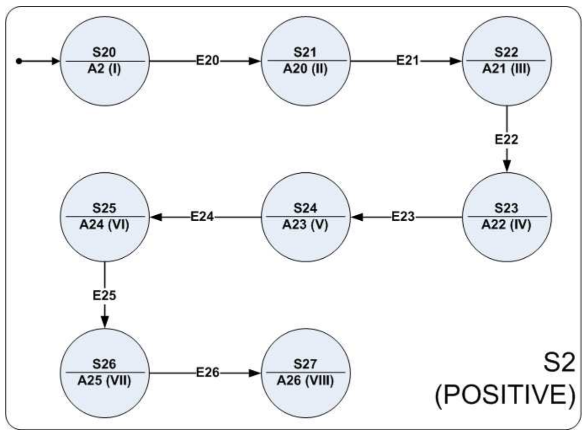


Fig. 12. FSM motion-generator state flow diagram – first sub-level

S20 (Fig. 12) includes the states for the first sector motion calculation (Table 5, and Fig. 13).

The door motion FSM was designed with Matlab/Simulink/StateFlow software. The StateFlow represents graphical FSM programming.

The StateFlow in the Simulink is presented as a block Door Motion FSM (Fig. 14). The FSM graphical programming is hidden in the block Door-Motion FSM (Fig. 15).

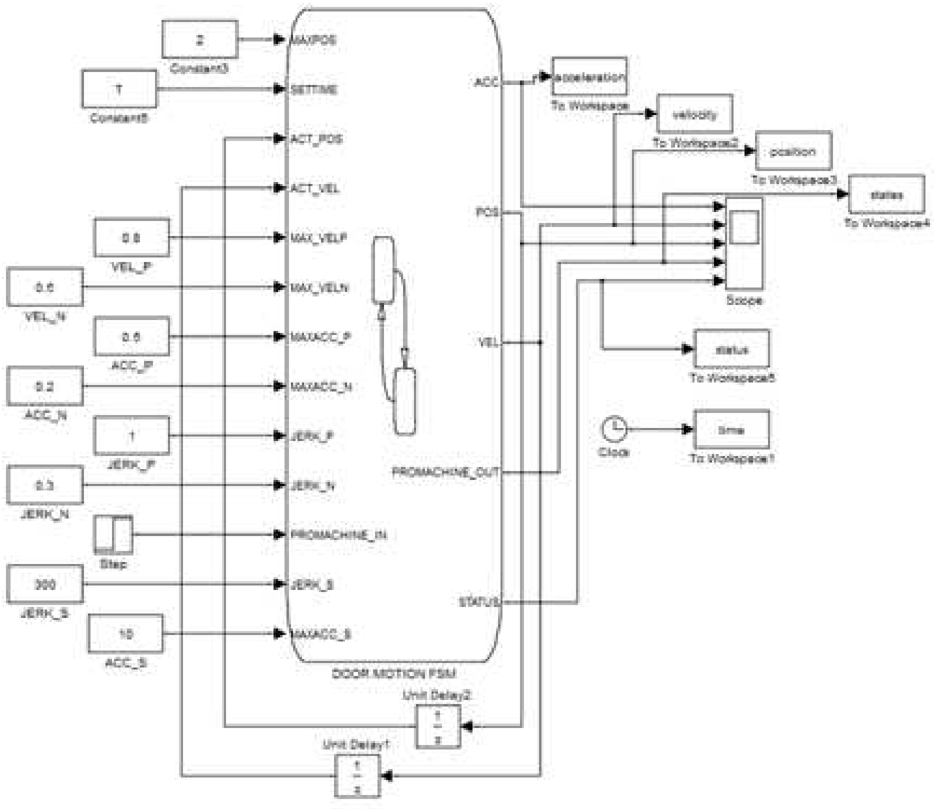
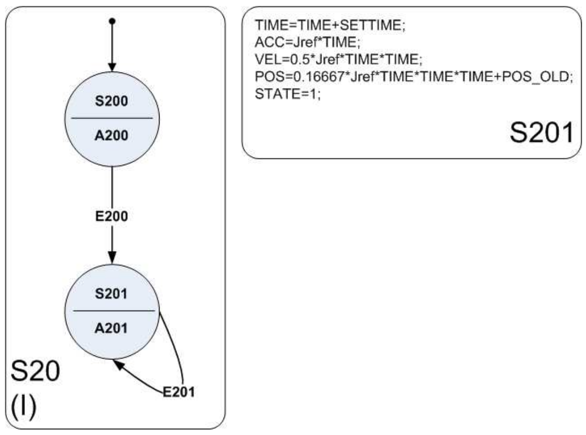


Fig. 14. Door-motion FSM block in Simulink



```
TIME=TIME+SETTIME;
ACC=Jref*TIME;
VEL=0.5*Jref*TIME*TIME;
POS=0.16667*Jref*TIME*TIME*TIME+POS_OLD;
STATE=1;
```

Fig. 13. FSM motion-generator state flow diagram – second sub-level

5 SELF-TUNNING ALGORITHM FOR MOTION GENERATOR

A main problem in all control systems is the physical limits (maximum current, acceleration, speed, etc.). The door-control limits are given within the power of a motor, and the door weight. These limits have to be considered when designing the control system. The control system is no longer controlled when the limits are achieved. As is known, the control system-task is there to reduce the margin of error between the actual and reference values. It is necessary to change the reference value so that the control system doesn't reach these limits. The door motion FSM includes changeable acceleration parameters (Fig. 16). The acceleration can be calculated with the help of Newton's second law (5). The force is a known parameter from a given motor data.

$$F_m - \hat{F}_f = \hat{m} \cdot a_{\max} \Rightarrow a_{\max} = \frac{F_m - \hat{F}_f}{\hat{m}} \qquad (5)$$

$$\hat{m} = \frac{\hat{F}_m}{\hat{a}_{ref}} = \frac{k_F \cdot i_m}{\hat{a}_{ref}} \qquad (6)$$

Linear motors have linear motion and already have the given motor power in force (Fm). The maximum acceleration can be calculated from motor force (Fm), estimated

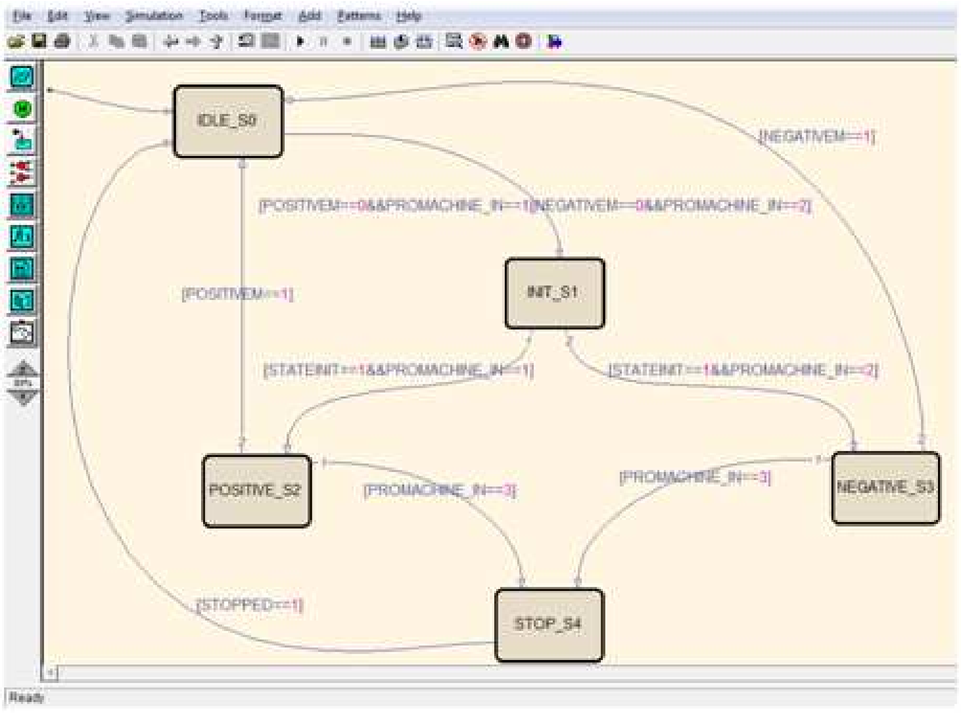

File Edit View Simulation Tools Format Add Patterns Help
IDLE_S0
[NEGATIVEM==1]
[POSITIVEM==0&&PROMACHINE_IN==1||NEGATIVEM==0&&PROMACHINE_IN==2]
INIT_S1
[POSITIVEM==1]
[STATEINIT==1&&PROMACHINE_IN==1]
[STATEINIT==1&&PROMACHINE_IN==2]
POSITIVE_S2
[PROMACHINE_IN==3]
[PROMACHINE_IN==3]
NEGATIVE_S3
[STOPPED==1]
STOP_S4
Ready

Fig. 15. Door motion FSM graphical programming in StateFlow

Events	Description
E20	Time t_1 expired
E21	Time t_2 expired
E22	Time t_3 expired
E23	Time t_4 expired
E24	Time t_5 expired
E25	Time t_6 expired
E26	Time t_7 expired
Actions	**Description**
A20	Execute sector II
A21	Execute sector III
A22	Execute sector IV
A23	Execute sector V
A24	Execute sector VI
A25	Execute sector VII
A26	Execute sector VIII

	Transition table							
Events States	E20	E21	E22	E23	E24	E25	E26	E4 E6
S20 (I)	A_{20}/S_{21}	-	-	-	-	-	-	A_4/S_4
S21 (II)	-	A_{21}/S_{22}	-	-	-	-	-	A_4/S_4
S22 (III)	-	-	A_{22}/S_{23}	-	-	-	-	A_4/S_4
S23 (IV)	-	-	-	A_{23}/S_{24}	-	-	-	A_4/S_4
S24 (V)	-	-	-	-	A_{24}/S_{25}	-	-	A_4/S_4
S25 (VI)	-	-	-	-	-	A_{25}/S_{26}	-	A_4/S_4
S26 (VII)	-	-	-	-	-	-	A_{26}/S_{27}	A_4/S_4
S27 (VIII)	-	-	-	-	-	-	-	A_0/S_0 A_4/S_4

Table 5. FSM motion-generator transition matrix – second sub-level

friction force (F_f), and estimated door mass (m). The estimated mass (6) can be calculated from the estimated force given from the motor (F_m) and the estimated reference acceleration (a_{ref}) given from the motion-generator. The motor estimated force (F_m) is calculated from the measured motor current (i_m) and the motor force constant (k_F). This calculation needs proper motion control and assurance that the motor is within the motion limits. The calculation execution is carried out only when the acceleration reference (a_{ref}) is equal to the maximum configured acceleration during initialization. So, this calculation needs the additional state called "initialization". The calculated approximated mass (m) includes friction, so it is important that the mechanism is in good condition. The maximum acceleration limit (a_{max}) can be calculated (7) without the estimated friction forces (F_f) because these frictions are already included within the approximated door mass (m).

$$a_{\max} = \frac{F_m}{\hat{m}} \tag{7}$$

Rotary motors have rotary motion, and motor power is given through torque. A typical door application is a rotary motor with gears, a belt, and two belt-pulleys. The motor force Fm has to be calculated in this case. The calculated motor force (8) depends on the gear ratio (i), motor torque (τ_m) and belt pulley radius (r).

$$\tau_m = \frac{r \cdot F_m}{i} \Rightarrow F_m = \frac{\tau_m \cdot i}{r} \tag{8}$$

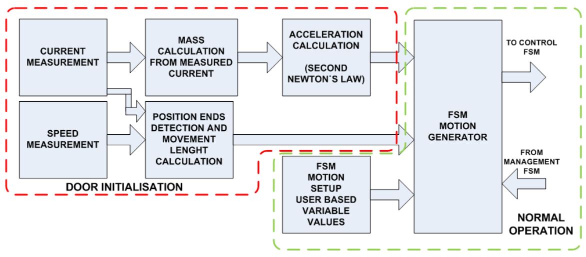

CURRENT MEASUREMENT
MASS CALCULATION FROM MEASURED CURRENT
ACCELERATION CALCULATION (SECOND NEWTON'S LAW)
SPEED MEASUREMENT
POSITION ENDS DETECTION AND MOVEMENT LENGHT CALCULATION
DOOR INITIALISATION
FSM MOTION SETUP USER BASED VARIABLE VALUES
FSM MOTION GENERATOR
TO CONTROL FSM
FROM MANAGEMENT FSM
NORMAL OPERATION

Fig. 16. Automatic parameters' calculation for FSM motion generator

We already know the motor force F_m parameter for both types of motors. An estimated mass (m) is the second parameter that is needed for acceleration calculation. The mass can be calculated from a motor measured current. The doors have an initiation cycle at the beginning (power on). The doors go into end-position search at a constant slow speed. The measured motor current increases at the beginning of the door's motion. The maximum motor current value is saved in the memory as the door-opening interval. The motor's maximum current increases when the door's weight is increased. The mass can be calculated from the measured maximum motor current and the mass constant. The mass constant is achieved using experimental measures on the sliding door application. The acceleration can be calculated now that both parameters are known. The door-motion FSM will generate reference motion values within the motor-capacity using a given door weight. The control system cannot achieve the limits using these types of door motion FSM.

The door during normal operation goes into blockade detection when the motion is forcefully interrupted (obstacle collision or any other movement prevention). In this case the door changes movement direction or stops with an error indicator after three subsequent attempts.

6 EXPERIMENTS

The experiment was executed using two methods. The first method represents the linear mechanism in a vertical position. This method includes the gravity influence on the load that increases the control complexity of the linear mechanism. The second method represents the linear mechanism in a horizontal position. The gravity influence on the load was excluded in this case. The control complexity is smaller than that by the vertical position of the linear mechanism. The control system was composed of the control loop with a PI estimator (load estimation), an S-shaped velocity generator, and the functionality-control algorithm. The motion was executed via an external push-button. The process measures and settings were processed via a DSP terminal on the PC, connected to the DSP system via the USB cable.

The first part of the experiment was executed with lower jerk ($2\text{m}^3/\text{s}$) and acceleration ($2\text{m}^2/\text{s}$). The position (0.3 m) and velocity (0.3 m/s) were constant. The motion generator calculated the motion limits using the given parameters of jerk, acceleration, velocity, and position. The MFSM motion generator started calculating the motion profile by a given event from the control management MFSM (Fig. 2).

The control-loop was designed with feed-back and feed-forward loops. The overall force (current reference) was the sum of the results from both loops. The feed-back loop eliminated disturbances by fast changes in motion by the given estimated load mass on the linear mechanism (by motion starts and ends, sudden external disturbances etc.). The feed-forward loop friction force (feed-forward current) was needed to overcome the mechanism friction and gravity effect. The linear mechanism was positioned in two ways as mentioned. The difference between two positions was visible from the force measures. The vertically-positioned mechanism (Fig. 18) had a lower friction force when the load moved downwards because of the gravity effect on the load. The friction force was much larger when the load moved upwards because the motor needed to overcome the gravity effect and mechanism friction. The friction force was balanced in both motion directions by the horizontal position of the mechanism. A small force difference appeared between directions because the frictions were different in both motion directions (Fig. 17).

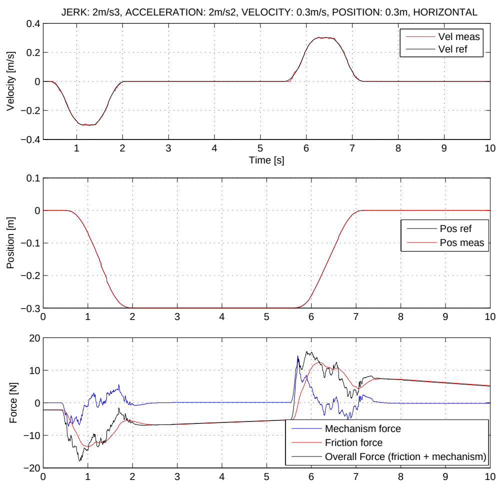


Fig. 17. Motion of the linear mechanism with given settings by horizontal position

The next experiment was executed with an increased jerk parameter of $10\text{m}^3/\text{s}$ (Fig. 19, Fig. 20). The results from the measures showed that the force peaks were

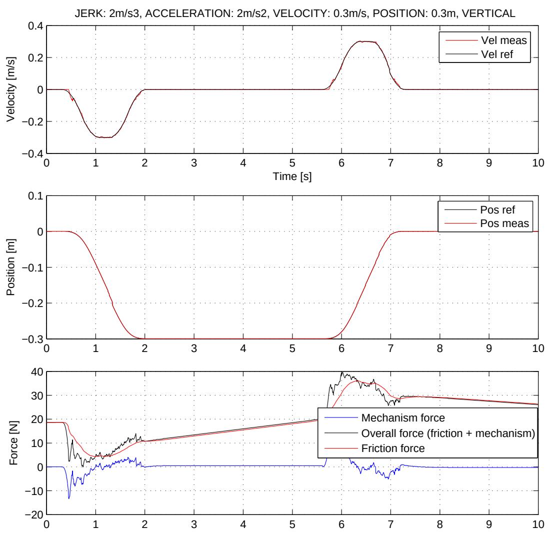


Fig. 18. *Motion of the linear mechanism with given settings by vertical position*

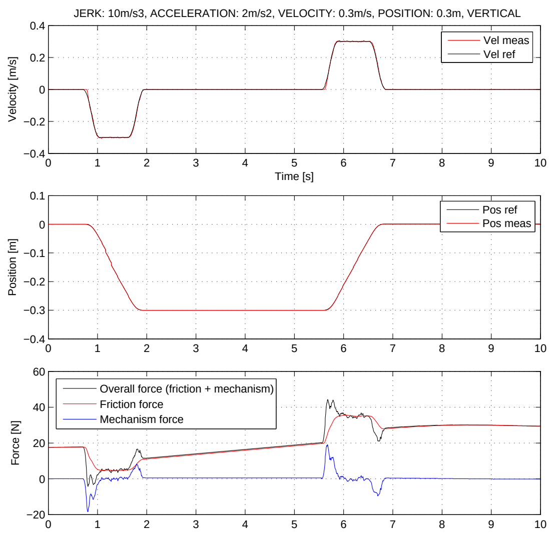


Fig. 20. *Motion of the linear mechanism with given settings by vertical position*

greater. The increased force peaks were logical because the motion dynamics increased with greater jerk or acceleration.

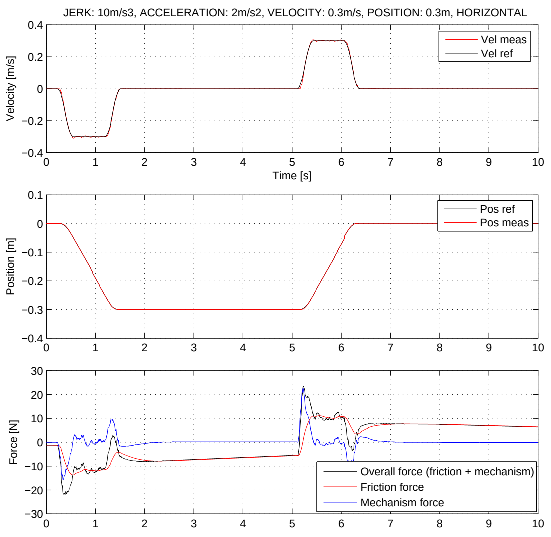


Fig. 19. *Motion of the linear mechanism with given settings by horizontal position*

The last experiment was executed with an increased jerk-parameter of $50\text{m}^3/2$, and an acceleration parameter of $10\text{m}^2/\text{s}$ (Fig. 21, Fig. 22). The velocity profile was similar to the trapezoidal, so the velocity crossings were rough. The jerk parameter was a minor variable for current peak-definition. The larger jerk (trapezoidal shapes) had the greatest force-peaks, so the power consumption, motor heating, and mechanism vibrations were greater from these types of motion profile shapes. It was very important to include such control systems that can be adjusted with the jerk-parameter. The motion dynamics could be adjusted with any change in load mass and current fluctuations with proper monitoring algorithms, which could be simply integrated within FSM. The current response showed all the needed information for manipulating motion dynamics for the best motion performance on chosen mechanisms.

7 CONCLUSION

Sequential S-shaped motion profile generation strategy with MFSM was proposed for reducing the motion-induced jerk, vibration, and program structure complexity. The motion shapes are adapted by the user parameter (maximum jerk, velocity, and position) and activated by event-based input. The MFSM motion generator was designed in the Matlab/Simulink/Flowchart software, which is based on the FSM for rapid prototyping. The motion generator was designed with the help of modified mathematical models of the velocity S profile. The experiments on the DSP system with a door mechanism were executed to ensure the functionalities of the proposed techniques. The proposed motion-generator improved the door motion control, energy consumption, decreases mechanism vibrations, etc. The generator reduced jerk at each motion start if the previous peak force exceeded the given user limit. Such a method eliminated hard vibrations on the mechanism. The motion reference calculations included deviation because of the algorithm's discretization. This devi-

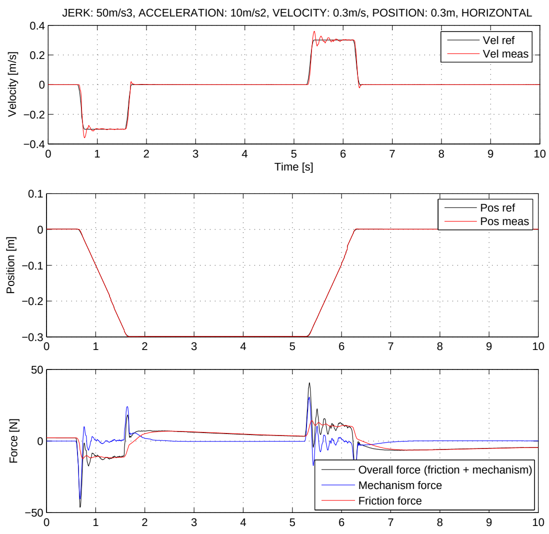


Fig. 21. Motion of the linear mechanism with given settings by horizontal position

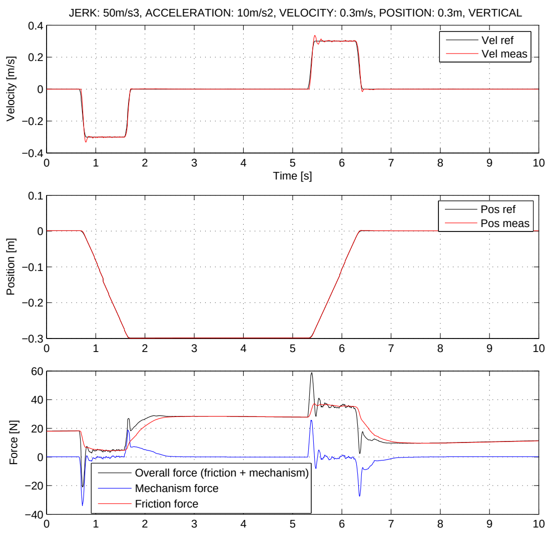


Fig. 22. Motion of the linear mechanism with given settings by vertical position

ation can be reduced with smaller time-steps during algorithm execution. These deviations didn't affect the door-motion too much because the application had imprecise control. The deviations of the motion references in the experiment were 2e-4 m/s for the velocity reference and 4e-4 m for the positional reference with an algorithm step-time of 200e-6. The variations in the motion references could be reduced with additional improvements in the motion calculations.

Speed-control was only included within the current door application. The motor current fluctuated, so a current-control was needed for control optimization that would reduce motor heating, energy consumption, and maintenance.

The motion vibration performance was based on jerk step decrease only. The future plan is to increase the performance on multiple parameters' tuning using a genetic algorithm and a fitness function [17].

ACKNOWLEDGMENT

Operation part financed by the European Union, European Social Fund. Operation implemented in the framework of the Operational Programme for Human Resources Development for the Period 2007-2013, Priority axis 1: Promoting entrepreneurship and adaptability, Main type of activity 1.1.: Experts and researchers for competitive enterprises.

REFERENCES

[1] K. Jezernik, A. *Hace, Hybrid system for event-based planning and control of robot operation*, Robotics in Alpe-Adria-Danube Region (RAAD), ISBN: 978-1-4244-6885-0, 2010.

[2] Ferdinand, W. et al., *Modeling Software with Finite State Machines – A Practical Approach*, Auerbach Publications, 2006 [5]

[3] Richard, F. T., *Asynchronous Sequential Machine Design and Analysis: A Comprehensive Development of the Design and Analysis of Clock-Independed State Machines and Systems*, Morgan&Claypool Publishers, 2009

[4] C.G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. Springer, 2008.

[5] P. J. Ramadge, W. M. Wonham, *Supervisory Control of a Class of Discrete Event Process, SIAM J.* Control and Optimization, vol. 25, no. 1, pp. 206-230, 1987.

[6] M. D. Natale, H. *Zeng, Task implementation of synchronous finite state machines,* Design, Automation & Test in Europe Conference & Exhibition, ISBN: 978-1-4577-2145-8, 2012.

[7] Kim, D. N. Et al., *Planning algorithms for s-curve trajectories,* Advanced intelligent mechatronics, IEE/ASME international conference, ISBN 978-1-4244-1263-1, 2007.

[8] S. Macfarlane, E. Croft, *A Jerk-Bounded Manipulator Trajectory Planning: Design for Real-Time Applications.* IEEE Transactions on Robotics and Automation, vol. 19, no.1, pp.42-52, 2003.

[9] H. Z. Li, et al., *A New Motion Control Approach for Jerk and Transient Vibration Suppression*, 2007.

[10] A. Polič, K. Jezernik, *Event-driven Control strategy for a Three Phase Inverter*, Industrial Electronics, IEEE International Symposium, vol.2, pp. 1442 – 1447, 2006.

[11] J. Kopjak, J. Kovacs, *Event-driven control program models running on embedded systems,* Applied Computational Intelligence and Informatics (SACI), 6th IEEE International Symposium, ISBN: 978-1-4244-9108-7, 2011.

[12] E. G. Bryan, *Control Logic Requirements for Complex Manufacturing Systems,* NSF Workshop on Logic Control for Manufacturing systems, 2000.

[13] Hercog, D., Curkovic, M., Edelbaher, G., Urlep, E., *Programming of the DSP2 board with the Matlab/Simulink,* Industrial Technology, IEEE International Conference, ISBN: 0-7803-7852-0, 2003.

[14] A. Rojko, K. Jezernik, *Disturbance rejection by PI estimator in position robot control,* Industrial Electronics, Proceedings of the IEEE International Symposium, vol.3, pp.1056 – 1061, 1999.

[15] D. Hercog, M. Curkovic, K. Jezernik, *DSP based rapid control prototyping systems for engineering education and research,* Computer Aided Control System Design, IEEE International Conference, ISBN: 0-7803-9797-5, 2006.

[16] G. H. Wang, et. al., *Research on pressure stabilized controller of hydraulic shaking table based on rapid prototyping of DSP algorithms using SIMULINK,* ISBN: 978-1-4244-8737-0, 2011.

[17] H. Zhang, Y. Cai, Y. Chen, *Parameter optimisation of PID controllers based on genetic algorithm,* International Conference on E-Health Networking, Digital Ecosystems and Technologies (EDT), vol. 1, pp.47-49, 2010.

Franc Hanžič received the B.S. degree in electrical engineering from the Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia, in 2008. He is currently a student of Ph.D degree as a young researcher in the Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia. His research interests include mechatronic systems, intelligent motion control, software design and ARM Cortex microcontrollers.

Karel Jezernik received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Ljubljana, Ljubljana, Slovenia, in 1968, 1974, and 1976, respectively. He was a Visiting Research Fellow at the Institute of Control, TU Braunschweig, during 1974-1975. In 1976, he joined the University of Maribor, Maribor, Slovenia, where, since 1985, he has been a Full Professor and Head of the Institute of Robotics. His research and teaching interests include automatic control, robotics, power electronics, mechatronics, and electrical drives. Prof. Jezernik is Vice President for Workshop Activities of the IEEE Industrial Electronics Society.

Slavko Cehner received the B.S. degree in electrical engineering from the Faculty of Electrical Engineering and Computer Science, University of Maribor, Maribor, Slovenia, in 1988. For some years he was working in development of ISKRA TENEL and ISKRA TEL Slovenia, mainly on automatic control, motor drive control and parallel processing in automation. He is currently leading a development as a general manager of company Doorson d.o.o., Maribor, Slovenia.

AUTHORS' ADDRESSES

Franc Hanžič, B.S.
Doorson d.o.o.
Milenkova ulica 9, SI-2000 Maribor, Slovenia
email: franc.hanzic@doorson.si
Prof. Karel Jezernik, Ph.D.
Institute for Robotics,
Faculty of Electrical Engineering and Computer Science,
University of Maribor,
Smetanova ulica 17, SI-2000 Maribor, Slovenia
email: karel.jezernik@uni-mb.si
Slavko Cehner, B.S.
Doorson d.o.o.
Milenkova ulica 9, SI-2000 Maribor, Slovenia
email: slavko.cehner@doorson.si

Received: 2012-06-18
Accepted: 2012-10-15