Development of Flight Control Panel Interface for Electronic Flight Control System of High Aspect Ratio Light Utility Aircraft

Eries Bagita Jayanti[a], Sayr Bahri[b], Novita Atmasari[c], Adi Wirawan[d]

National Institute of Aeronautics and Space (LAPAN), Indonesia

[a]Corresponding author: eries.bagita@lapan.go.id
[b]sayr.bahri@lapan.go.id
[c]novita.atmasari@lapan.go.id
[d]adi.wirawan@lapan.go.id

Abstract. LAPAN Surveillance Aircraft (LSA) - 02 is a high aspect ratio light utility aircraft that is supplemented with Electronic Flight Control System (EFCS) to perform as an Unmanned Aerial Vehicle (UAV) technology demonstrator. To perform its mission, the test pilot onboard has to input commands via Human Machine Interface (HMI) namely Flight Control Panel (FCP) which is a part of EFCS. The FCP has three main functions which are to load and send initialization data to Flight Control Computer (FCC), to allow the pilot to enter data or give commands manually to FCC, and to display information of sensor, modes, and vice versa to the pilot. However, FCP is merely an input-output device that does not contain algorithms or functions related to aircraft and its flight control system. Therefore, it is required to develop Flight Control Panel Interface (FCPI) to interpret signals from buttons and knobs on FCP, convert and condition those signals to Flight Control Laws (FCLs), and feeds back those signals and other information such as modes and sensors to FCP. The research in this paper shows the development of FCPI which is integrated into the previously developed FCLs model. The development process adopts V model method which starts from the definition of requirements, design of software, and verification by means of EFCS simulation. The results are the refinement of EFCS architecture and FCPI is developed for autopilot functions such as airspeed, heading, altitude, and vertical speed.

INTRODUCTION

Electronic Flight Control System (EFCS) is developed for LAPAN Surveillance Aircraft (LSA) – 02 to perform UAV Technologies. LSA – 02 uses a high aspect ratio light utility aircraft namely STEMME ES-15 which is manufactured in Germany shown as in FIGURE 1. The EFCS is then supplemented to the basic aircraft and transforms the conventional mechanically controlled aircraft into an automatically controlled aircraft. Maneuvers performed during the EFCS mode is realized by using Flight Control Laws (FCLs)[1–3]. FCLs software is planted inside Flight Control Computer (FCC). It receives signals from Flight Control Panel (FCP) and sensor and produces signals to actuators and FCP[4,5]. FCP is operated mainly by a test pilot on board. FCP can be optionally operated from the ground control station. The basic function developed for FCP is autopilot mode which consists of airspeed control, heading control, altitude control, and vertical speed control[6–8]. FCP is placed on the avionics panel in front of the pilot and is the main interface between the test pilot and EFCS of the LSA-02 Technology Demonstrator.

The FCP is developed inside research cooperation between LAPAN Indonesia and TU Berlin Germany by LAPAN's requirements. The device displays information on the 7" TFT display which is protected by the shatterproof glass as a safety feature[6]. The pilot gives inputs through the buttons and knobs which are located surrounding the display. FCP has three main functions which are to load and send initialization data to the FCC to be read by FCLs, to allow the test pilots to enter data or give commands manually to EFCS, and to display information to the test pilots[6,8]. In the process of FCP giving or getting information from FCLs, there is Flight Control Panel Interface (FCPI) which acts as a link between FCP and FCLs.

The reason of FCPI exists that FCP is designed only as an input-output device which means it is not allowed to be equipped with algorithms and functions which are related to aircraft and its flight control system. Therefore, it is essential to construct FCPI to realize these algorithms and functions. FCPI plays an important role to interpret signals generated from FCP which is formed in CAN communication, to convert and condition those signals to FCLs, and to feedback those signals and other information such as modes and sensors returning to FCP. This paper discusses research about the development of FCPI by following a development process called V model. The V model represents the relationships between each phase of the development life cycle and its associated phase of testing and verification. This research is performed by stages such as defining the concept and requirements of FCPI, constructing architecture, formulas, and design of FCPI, integrating the FCPI model to its parent models that are EFCS and aircraft, and performing FCPI verification with respect to the defined requirements.

FIGURE 1. STEMME ES-15

The objectives of this research are to develop FCPI in the Model in the Loop (MIL) level and to perform verification tests for FCPI design. The development of the FCPI in this paper is focused on implementing defined autopilot functions which are airspeed control, heading control, altitude control, vertical speed control, and vertical mode select.

LITERATURE REVIEW

V Model

The development process of the FCPI follows the V model[9]. The left leg of the V shape represents the evolution of user requirements into smaller components through the process of decomposition and definition in detailed design. On the other hand, the right leg represents the integration and verification of the system components. For the development process, the requirements shall be firstly defined, which are derived from the upper level requirements[9–11]. Writing the requirements, FCPI design shall have proceeded. Finally, the FCPI codes shall be verified against the written requirements. The illustration of V model for the development process of FCPI is shown in FIGURE 2.

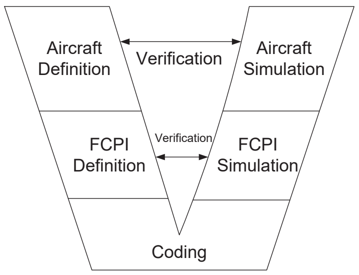


FIGURE 2. The V model of FCPI development

The V model consists of the number of development steps such as aircraft and FCPI concepts, requirements, specifications definition, FCPI coding, FCPI, and aircraft integration and verification by means of simulations. The work of concepts, requirements, and specifications definition has the purpose to arrange preparation prior to design. This work may occur to cycle internally and iteratively in each step. Here, the aircraft concept, requirements, and specifications are given from an upper level document. Later, the FCPI concept and requirements definition is performed. The definition includes, for example, the concept of FCPI software architecture and writing requirements of input-output variables. After that, the process continues to FCPI software block diagram construction, logic and equations formulation, and software coding. The next process is to integrate parameters and variables to FCPI software and run a verification test. Completing the FCPI simulation, the FCPI model is subsequently integrated into the EFCS model which is connected to the aircraft model. Here, aircraft simulation is conducted to perform final verification tests for FCPI development.

Architecture of EFCS

EFCS is developed to couple to the aircraft basic mechanical control system. This system is able to take over the aircraft control via an electro-mechanical actuation system that moves the basic mechanical control linkages. The aircraft maneuver is controlled by FCLs with commands come from the test pilot or optionally from the ground control station. EFCS supports automatic flight control mode which is activated by the test pilot when sending input commands to EFCS via FCP. The EFCS in automatic control mode can control the aircraft in all axes in longitudinal and lateral motions.

EFCS consists of two main parts which are the Basic Electronic Flight Control System (BEFCS) and Experimental System (XS). BEFCS provides the basic control functions which are computed by FCLs. BEFCS provides limited authority for XS to run experimental flight mode. BEFCS contains FCP, sensors, FCC, and actuators. XS is an additional system that is linked to the BEFCS via FCC. The XS can control aircraft via selected entry points which are regulated by FCC to limit XS authority. The XS receives a sensor signal from BEFCS and it is possible to install experimental payloads such as experimental sensors, radar, or camera. The XS reliability is not the main goal but to perform experimental functions as it is already monitored and controlled by reliable BEFCS. The architecture of EFCS is shown in FIGURE 3.

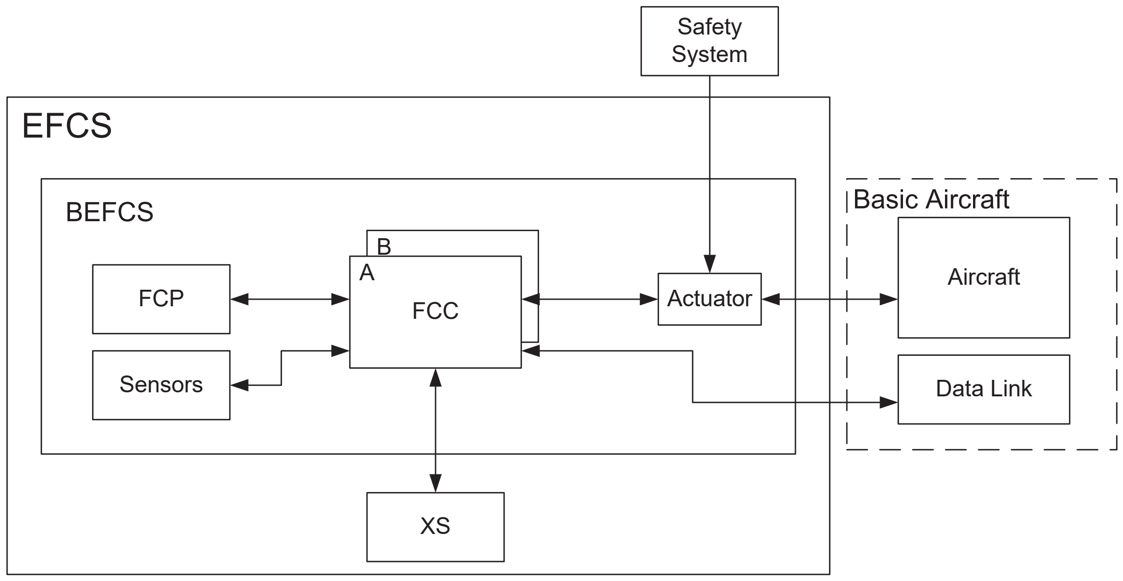


FIGURE 3. Architecture of EFCS

Architecture and Concept of FCP

FCP is the main human-machine interface (HMI) between the test pilot and EFCS of LSA-02 technology demonstrator aircraft[6]. The FCP which is placed on the avionics panel in front of the test pilot has three main functions i.e. to load and send initialization data to the FCC to be read by the FCLs, to enable the test pilot to enter data or give commands manually to the EFCS, and to display information of sensors, modes, and vice versa to the test pilot. The architecture of FCP and the allocation of buttons and knobs are set in FIGURE 4[12]. The figure shows the names of the buttons and knobs. The button's name is stated as "PB" which means push button which is followed by a unique number. While "R" stands for rotary switch or knobs which is also followed by a unique number[12]. The FCP provides in total 29 buttons and 6 rotary switches/knobs. The words in the second and third-row under buttons and knobs names represent the unique output data every time they are activated by pressing or turning[12]. The output data is listed in TABLE 1.

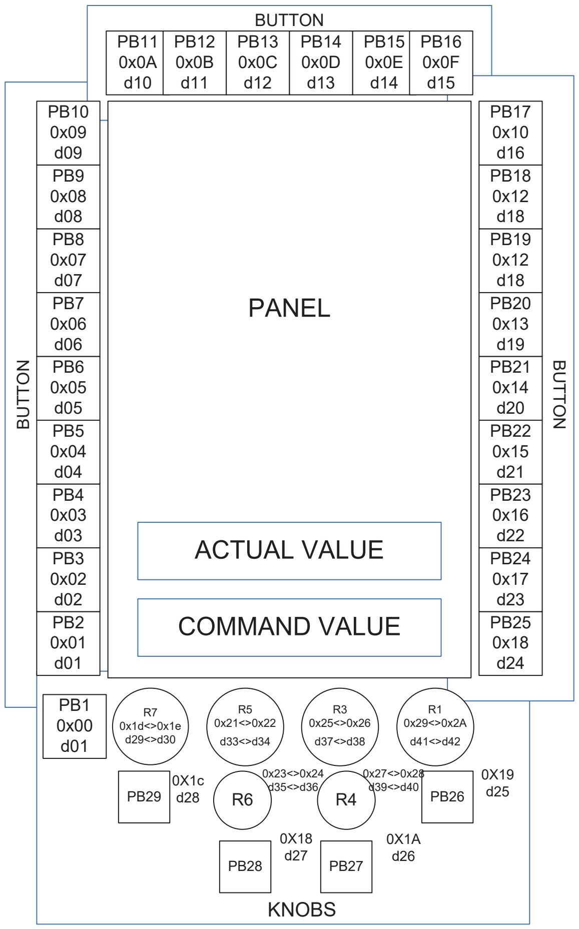


FIGURE 4. General Architecture of FCP drawn with blue line boxes. Button and Knobs identities are uniquely defined

TABLE 1. List of FCP keyboard data

Name	Data (Hexadecimal)	Data (Decimal)
Rot1L	0x29	41
Rot1R	0x2a	42
Rot3L	0x25	37
Rot3R	0x26	38
Rot4L	0x27	39
Rot4R	0x28	40
Rot5L	0x21	33
Rot5R	0x22	34
Rot6L	0x23	35
Rot6R	0x24	36
Rot7L	0x1d	29
Rot7R	0x1e	30
PB26	0x19	25
PB27	0x1b	26
PB28	0x1a	27
PB29	0x1c	28

RESULTS

Requirements of FCPI

The requirements definition begins with constructing the concept of EFCS phases which are shown in FIGURE 5. The EFCS phases occur inside the aircraft phase which contains take-off, climb, cruise, descent, and land. Aircraft systems including EFCS are turned on before starting to take-off. Here, EFCS is on but disengage which means the aircraft is still in manual flight mode. During this phase, FCC inside EFCS only monitors the aircraft maneuvers through EFCS sensors. After the aircraft is in the cruise phase, the test pilot is allowed to engage EFCS which causes the aircraft to convert from manual flight mode to electronic flight mode. However, during the EFCS engagement, the EFCS only maintains the previous steady flight condition. Right after the test pilot engages FCP and enters commands through buttons and knobs on FCP, the EFCS controls the aircraft's motion with respect to given commands. Completing the mission, the test pilot is then allowed to disengage FCP and subsequently disengage EFCS which returns the aircraft from electronic flight mode to manual flight mode. From there, the pilot can proceed to the next aircraft phases which are descent and land.

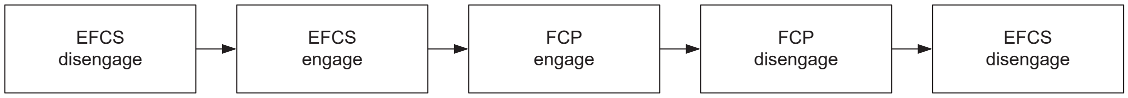


FIGURE 5. EFCS Phases

After FCP is engaged, the EFCS still maintains the previous steady flight condition. However, the test pilot now is able to enter specific commands via FCP. FCP shall show the sensor's measurement through actual value indicators. Note that the current developed autopilot mode covers only airspeed, heading, altitude, vertical speed, and vertical mode. Therefore, there are four actual value indicators plus a vertical mode indicator. In addition, the command value indicators placed under the corresponding actual value indicators show the commanded values taken from previous steady flight condition. The test pilot shall examine the vertical mode indicator and push the vertical mode button to go to the next vertical mode. The altitude control and the vertical speed control cannot run simultaneously so that it needs to be selected which control needs to execute via the vertical mode button. The default vertical mode is altitude and the other mode is vertical speed. Subsequently, the test pilot shall turn the knobs for changing desired commands for airspeed, heading, and altitude or vertical speed. By turning the knobs, the test pilot selects the precommand value. To enable an efficient input method, knobs for heading and altitude are equipped with coarse and fine-tunes. The pre command values are displayed in command value indicators. The pre command value limiters are activated when knobs are turned continuously to excess permitted commands. When the displayed precommand values are matched with the desired values, the test pilots shall activate these precommand values to command values by pressing the knobs.

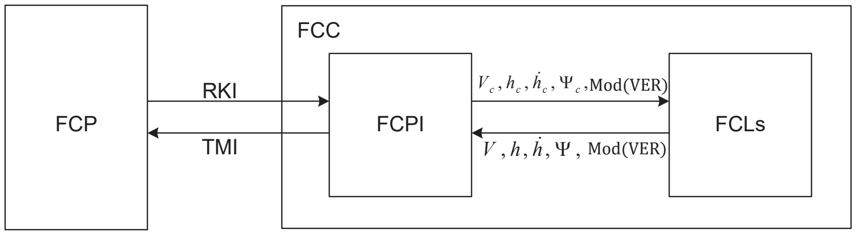

FCC
FCP
RKI
TMI
FCPI
$V_c, h_c, \dot{h}_c, \Psi_c$, Mod(VER)
$V, h, \dot{h}, \Psi$, Mod(VER)
FCLs

FIGURE 6. Block diagram of FCP and FCC

FCP is merely an input-output device that infers that there are no algorithms and functions related to aircraft and its flight control system exist. FCPI is developed to cover these algorithms and functions so that FCC is able to interpret precommand values set up when knobs are turned, to detect command values when knobs and vertical mode button are pressed. In addition, FCC can limit excessive inputs from knobs when they are violating predefined commands protection. Furthermore, FCC is able to send information of sensors measurement and commands to FCP. On the other hand, the commands activated via knobs are transmitted by FCPI to FCLs so that the FCLs can realize the aircraft maneuvers based on given commands. The working mechanism of FCP and FCC where FCPI and FCLs are developed is displayed in FIGURE 6.

FCPI acts as a control link between FCP and FCLs. FCPI shall be divided in two functionality components which are commands receiver and display transmitter which is called FCPI Command (FCPIC) and FCPI Display (FCPID) respectively. FCPIC processes physical interpretation for turning knobs and pressing knobs and vertical mode button sets protection limits for precommand inputs, detects commands activation from knobs pressing, and transmits these commands to FCLs. FCPID gathers preset precommand inputs values and current sensors information and converts them into American Standard Code for Information Interchange (ASCII) signals type prior to transmitting these signals to FCP. FCPI shall be equipped with parameters of push buttons and knobs unique identification number which is called as Receive Key Identification (RKI).

FCPI consists of some numbers of functions such as read function, limit function, activate function, and ASCII conversion function. Read function shall be able to determine if a knob is turned clockwise or counter clockwise for how many steps which represent precommand increment, to obtain precommand reference which is initially obtained from previous steady flight condition, and to compute final real precommand value. In addition read function shall be able to detect if the vertical mode button is pressed, to cycle vertical mode value when the button is pressed again, and transmit vertical mode input to FCLs. The limit function shall be able to limit inputted precommand value. Therefore, the limit function is provided with unique permissible ranges command values for airspeed, heading, altitude, and vertical speed. Activate function shall be able to detect if knobs and vertical mode buttons are pressed. Each command has its unique activation RKI. Receiving the activation RKI, the activate function takes the last given precommand value to become a command value and transmit it to FCLs. ASCII conversion function has purpose to convert precommand values including the vertical mode and associated sensor values from decimal signal type to ASCII signal type. During its process, there may be also unit conversions such as kilometer an hour from meter per second and degree from radian. The FCP is designed to receive ASCII signal type to be displayed on the FCP screen as decimal signal type. Here, ASCII conversion function transmits 6 bytes of signals which are called Transmit Message Identification (TMI) for each information of precommand values and sensor values. The ASCII conversion unit also ensures that each 6 information goes to the assigned actual value, command value, or vertical mode indicators.

After constructing the concept of FCPI contained architecture and functionalities and also deriving from upper-level requirements, the FCPI requirements are written carefully and its summary is listed in TABLE 2.

TABLE 2. Requirements of FCPI

No.	Title	Description
1.	FCPI component	FCPI Command (FCPIC) and FCPI Display (FCPID)
2.	FCPIC part	Airspeed, heading, altitude, vertical speed, vertical mode interface command
3.	FCPID part	Precommand input and sensor interface
4.	Interface command item	Read, limit, activate functions
5.	Interface display item	ASCII convert functions
6.	Button and knobs selection	Button for vertical mode Knobs for airspeed and vertical speed Double Knobs for heading and altitude
7.	Changing input values	Turn left (counter clockwise) to decrease and turn right (clockwise) to increase.
8.	Value changes increment	Airspeed $\pm 1\ km/h$, fine heading $\pm 1\ °$, coarse heading $\pm 10\ °$, fine altitude $\pm 10\ m$, coarse altitude $\pm 100\ m$, vertical speed $\pm 0.5\ m/s$.
9.	Vertical mode	Altitude mode = 1, Vertical speed mode = 2.
10.	Read Function	Identify RKI and increment based on trim point reference
11.	Limit function	Airspeed precommand ($130 \le V_{cip,kmh} \le 190$), Heading precommand ($0 \le \Psi_{cip,deg} < 360$), Altitude precommand ($1000 \le \mathrm{h}_{cip} \le 3000$), Vertical speed precommand ($-3.5 \le \dot{h}_{cip} \le 5$)
12.	Activate Function	Activate precommand to command input.
13.	Actual value	Display sensor value of Airspeed V_{kmh}, heading Ψ_{deg}, altitude h, and vertical speed $\dot{h}$
14.	Command value	Display precommand for Airspeed $V_{cip,kmh}$, heading $\Psi_{cip,deg}$, altitude h_{cip}, and vertical speed $\dot{h}_{cip}$
15.	Vertical mode value	Display status for vertical mode value
16.	Unit converter	Convert to SI unit for computation of logics and formulas.
17.	ASCII converter	Convert to ASCII for TMI

Design of FCPI

Based on written requirements, FCPI block diagram is constructed. Signal flows two way between FCP and FCC but here, it is graphically separated so that the signal flows as only in one direction from left to right. FCP is then divided into FCP Command (FCPC) and FCP Display (FCPD). FCPI is also separated based on its functionality, i.e. FCPIC and FCPID. FCPIC receives RKI signals to FCPC while FCPID transmits TMI signals to FCPD. FCPIC consists of read function, limit function, and activate function while FCPID comprises ASCII convert function. The block diagram of FCPI is shown in FIGURE 7.

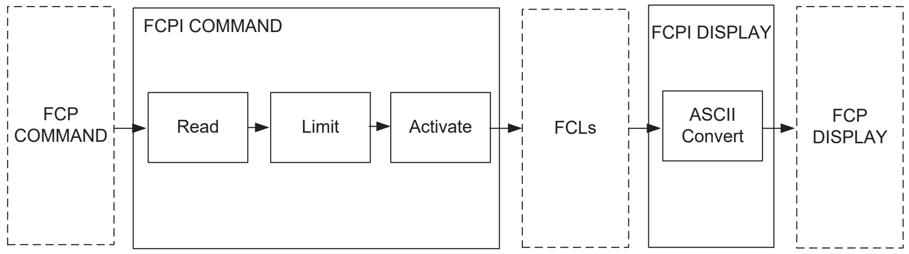


FIGURE 7. Block diagram of FCPI which is divided to FCPI Command (FCPIC) and FCPI Display (FCPID)

Read Function

Read function basically recognizes RKI for specific precommand either airspeed, heading, altitude, vertical speed or vertical mode commands. Airspeed and vertical speed consist of RKI for increase by turning clockwise knob, decrease by turning counterclockwise knob, and activate by pressing knob. Heading and altitude RKI have similar behavior but they are extended to be coarse increase by turning clockwise outer knob, fine increase by turning clockwise inner knob, coarse decrease turning counterclockwise outer knob, fine decrease by turning counterclockwise inner knob, and activate by pressing the knob. Vertical mode only consists of RKI for increase. However, since there are only two values for vertical mode, i.e. 1 for altitude which is set as default and 2 for vertical speed then pressing vertical mode button will set vertical mode from 1 to 2 and pressing the button again will cycle the vertical mode from 2 to 1. There is no activate RKI for vertical mode which means every time vertical mode button is pressed the vertical mode changes from one altitude control to vertical speed control or other way around. Every time increase or decrease RKI for specific precommand is pressed, the read function counts and saves number of steps resulted from turning the knob. In addition, the read function shall get the reference precommand which is obtained from the previous steady flight conditions with respect to the specific precommand. Without current reference, read function will always start from zero which creates faulty maneuvers especially for airspeed and altitude. The block diagram of read function is depicted in FIGURE 8.

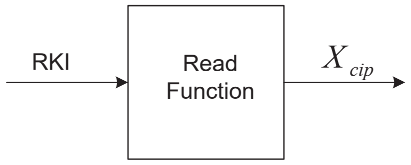


FIGURE 8. Read Function

Read functions for airspeed and vertical speed contains algorithms which can be written as follows

$$X_{cip}\begin{cases} = X_{cip,pr} + X_{cir} - \delta X_{cip} & ,RKI\ for\ decrease \\ = X_{cip,pr} + X_{cir} + \delta X_{cip} & ,RKI\ for\ increase \\ \quad X_{cip} & ,No\ RKI \end{cases} \tag{1}$$

with $X = V, \dot{h}$, index "cip, pr " stands for previous precommand input, index "cir" stands for command input reference, and index "cip" stands for precommand input. Note that X_{cip}in right hand side is obtained from sensor measurement at t_0, and save it to memory to be maintained for following iteration time. When new X_{cip} is yielded, the right hand side X_{cip} is updated in the next iteration time. In addition, the read functions for heading and altitude contains slightly more complicated algorithms since they have coarse and fine decrease and increase. The algorithm can be seen in following

$$X_{cip} \begin{cases} X_{cip,pr} + X_{cir} - \delta X_{cip,cr} & ,RKI\ for\ coarse\ decrease \\ X_{cip,pr} + X_{cir} + \delta X_{cip,cr} & RKI\ for\ coarse\ incrase \\ X_{cip,pr} + X_{cir} - \delta X_{cip,fn} & ,RKI\ for\ fine\ decrease \\ X_{cip,pr} + X_{cir} + \delta X_{cip,fn} & ,RKI\ for\ fine\ increase \\ X_{cip} & ,No\ RKI \end{cases} \tag{2}$$

with $X = \Psi, h$. Index "cip, pr" stands for previous precommand input, index "cir" stands for command input reference, index "cip,cr" stands for coarse precommand input, index "cip,fn" stands for fine precommand input, and index "cip" stands for precommand input. Note that X_{cip}in right hand side is obtained from sensor measurement at t_0, and save it to memory to be maintained for following iteration time. When new X_{cip} is yielded, the right hand side X_{cip} is updated in the next iteration time. Finally, the read function for vertical mode is more simply and is written to be following.

$$Mod(VER) \begin{cases} = Mod(VER)_{pr} + \delta Mod(VER) & ,RKI\ for\ increase \\ = Mod(VER)_{pr} & ,No\ RKI \end{cases} \tag{3}$$

with Mod(VER) stands for vertical mode, index "pr" stands for previous.

Limit Function

Limit function checks if incoming precommand does not exceed boundaries of permissible precommand values. Violating the maximum permissible precommand value, the given precommand shall be set at maximum limit while in case of exceeding minimum permissible precommand value, the given precommand shall be set at minimum limit. Block diagram for limit function is shown as FIGURE 9 and logic for limit function is written in equation 4.

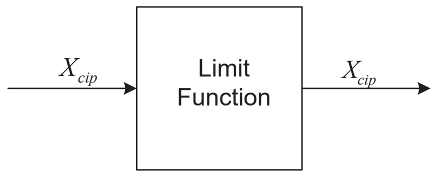


FIGURE 9. Limit Function

$$X_{cip} \begin{cases} = X_{cip,mn}, & if\ X_{cip} < X_{cip,mn} \\ = X_{cip,mx}, & if\ X_{cip} > X_{cip,mx} \end{cases} \tag{4}$$

with $X = V, \Psi, h, \dot{h}$. Index "cip" stands for precommand input, index "cip,mn" stands for minimum precommand input, and index "cip,mx" stands for maximum precommand input. For heading command limit function, it has purpose to cycle command from zero to less than 2π or 360° which can be realized by implementing following equation.

$$\Psi_{cip,o} = mod(\Psi_{cip,i}, 2\pi) \tag{5}$$

with index "cip,o" stands for output precommand input and index "cip,i" stands for input precommand input. Vertical mode limit function is realized by cycle the previous RKI to the next RKI, i.e. from 1 to 2 and from 2 to 1. The algorithm is given to be following.

$$Mod(VER)_o \begin{cases} = 1 & ,Mod(VER)_i > 2 \\ = Mod(VER)_i & ,Mod(VER)_i \leq 2 \end{cases} \tag{6}$$

Activate Function

Activate function simply activates specific precommand input into command input. This function is available only to airspeed, heading, altitude, and vertical speed. The vertical mode does not need to possess activate function as selection to vertical mode shall go the next value of vertical mode. The activate function is depicted in FIGURE 10.

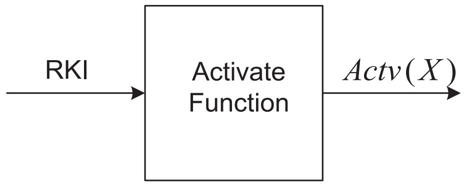


FIGURE 10. Activate Function

Activate function is defined in the following equation.

$$Actv(X)\begin{cases}=1 & ,RKI\ for\ activate\\ =0 & ,No\ RKI\end{cases} \tag{7}$$

with $X = V, \Psi, h, \dot{h}$.

Verification of FCPI

The verification of FCPI is conducted to evaluate if the requirements of the FCPI are correctly implemented. Verification is done both in architecture and functionality. The architecture consists of block diagrams which comprise functions such as read, limiter, activate and ASCII converter. The implementation of architecture and functions is done by means of MATLAB/Simulink.

Architecture

The FCPI consists of two components, i.e. FCPIC and FCPID which have separate purpose to receive RKI and to transmit TMI respectively. The implementation of FCPIC and FCPID is depicted in FIGURE 11 and FIGURE **12**.

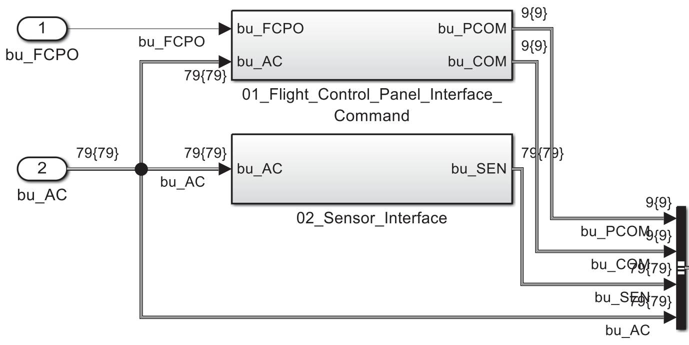
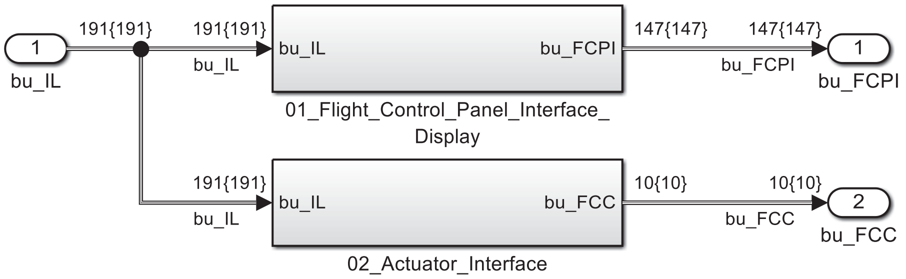


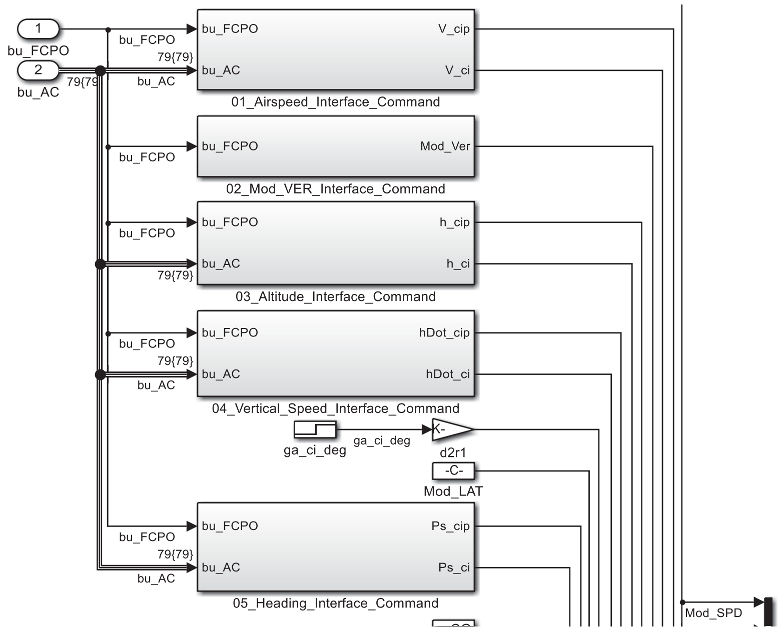
FIGURE 11. FCPIC implementation in Simulink model



.

FIGURE 12. FCPID implementation in Simulink model

Following architecture is availability of FCPIC part which consists of airspeed, heading, altitude, vertical speed, and vertical mode interface command. On the other hand, FCPID part contains precommand input and sensor interface. The FCPIC and FCPID parts verification is displayed in FIGURE 13 and FIGURE 14.



bu_FCPO
bu_AC
01_Airspeed_Interface_Command
V_cip
V_ci
02_Mod_VER_Interface_Command
Mod_Ver
03_Altitude_Interface_Command
h_cip
h_ci
04_Vertical_Speed_Interface_Command
hDot_cip
hDot_ci
ga_ci_deg
d2r1
Mod_LAT
05_Heading_Interface_Command
Ps_cip
Ps_ci
Mod_SPD
79{79}

FIGURE 13. FCPIC part implementation in Simulink model.

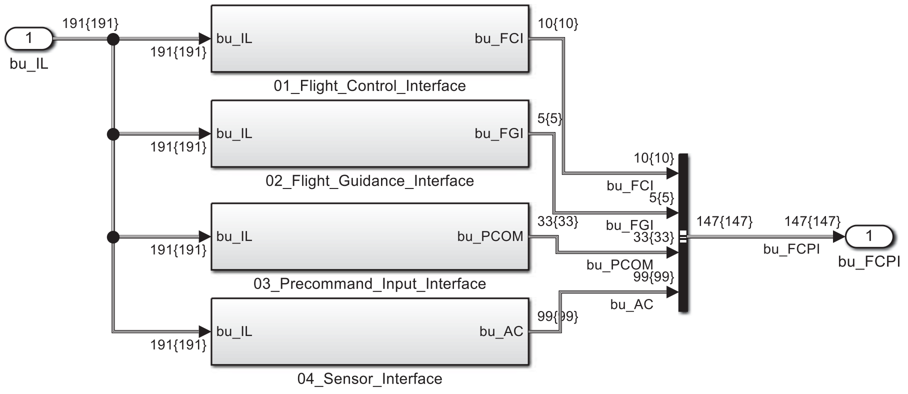


FIGURE 14. FCPID part implementation in Simulink model.

In each airspeed, heading, altitude, vertical speed interface commands shall be constructed with read, limit and activate functions while for the vertical mode shall only consists of read and limit function. Units for airspeed and heading prior entering read function are converted to meter per second and radian respectively. The implementation for these functions availability are depicted in FIGURE 15 and FIGURE **16**.

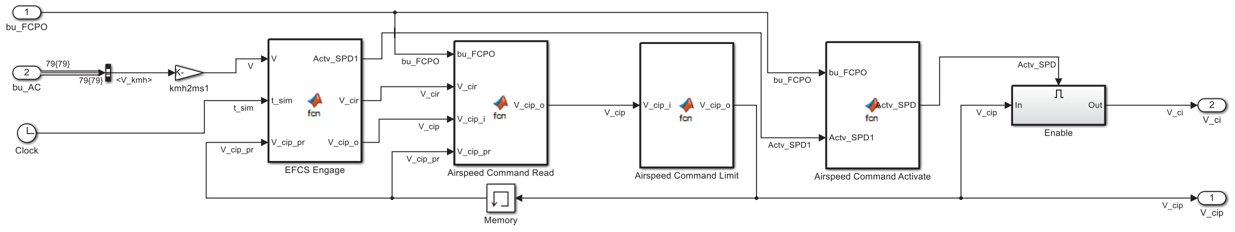


FIGURE 15. Example for read, limit and active functions for airspeed command

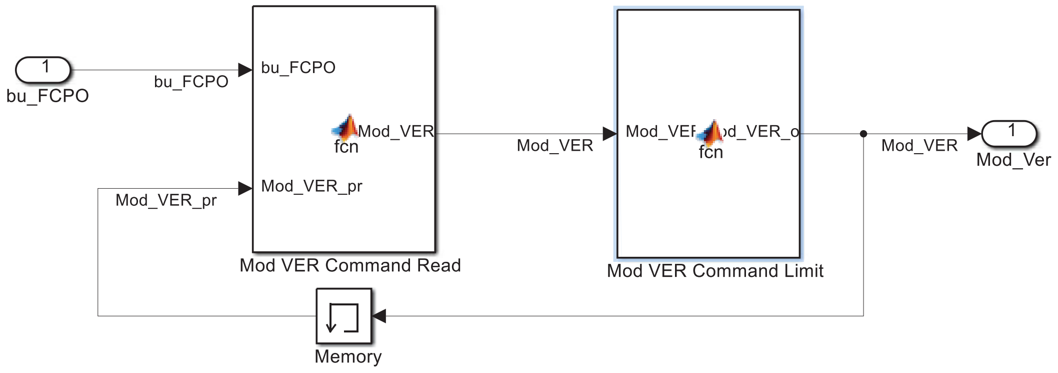


FIGURE 16. Read and limit functions for vertical mode command

On the other hand, interface display only possesses ASCII converter both for precommand input and sensor interface. Units of precommand input interface for airspeed and heading are converted to kilometer an hour and degree respectively. The implementation for ASCII convert function is given in FIGURE 17.

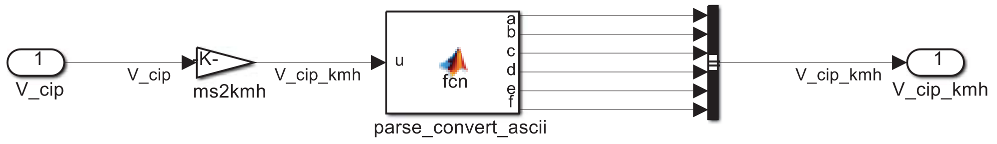


FIGURE 17. Example of ASCII convert function in airspeed precommand input

Functions

The FCPI model consisted of FCPIC and FCPID components is integrated to FCC model inside EFCS model. The FCPIC is integrated to input interface of FCC while FCPID is integrated to output interface of FCC. The EFCS is connected to aircraft model in order to run aircraft simulation for FCPI functionality verification purpose. The integration of FCPI to FCC and the connection between EFCS to aircraft are displayed in FIGURE 18 and FIGURE 19.

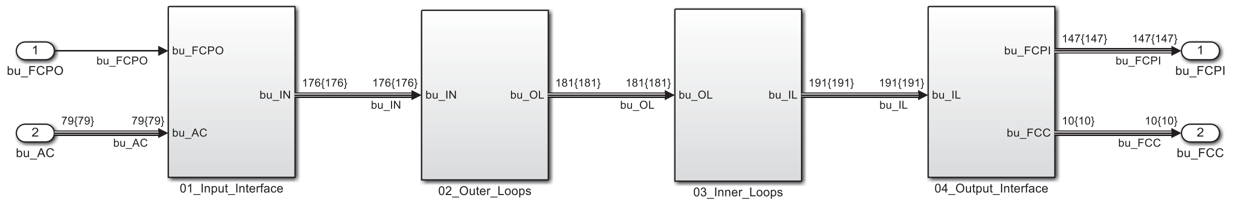


FIGURE 18. Integration FCPI to FCC

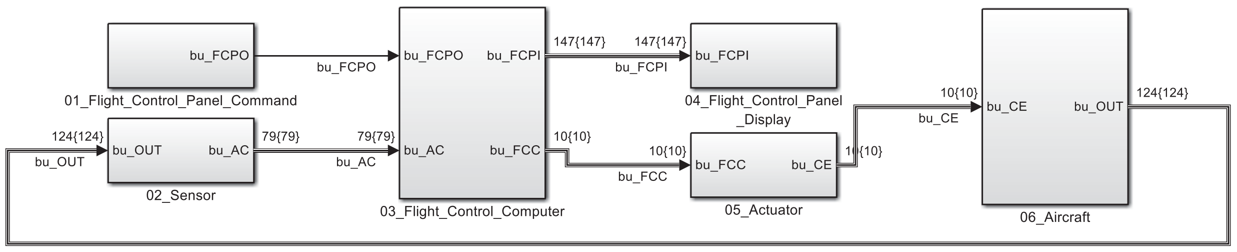

01_Flight_Control_Panel_Command
02_Sensor
03_Flight_Control_Computer
04_Flight_Control_Panel_Display
05_Actuator
06_Aircraft

FIGURE 19. Connection of EFCS to aircraft

Prior to functionality verification, tests definition is set up. The tests definition has purpose to plan the verification tests in order to examine if FCPI fulfills the written requirements specifically for changing input values, value changes increment, vertical mode, read function, limit function, activate function, actual value, command value, and vertical mode value. These requirements have been already stated in TABLE 2. In this paper, the tests definition discussed is focused on two tests, i.e. the verification for precommand and verification for vertical mode.

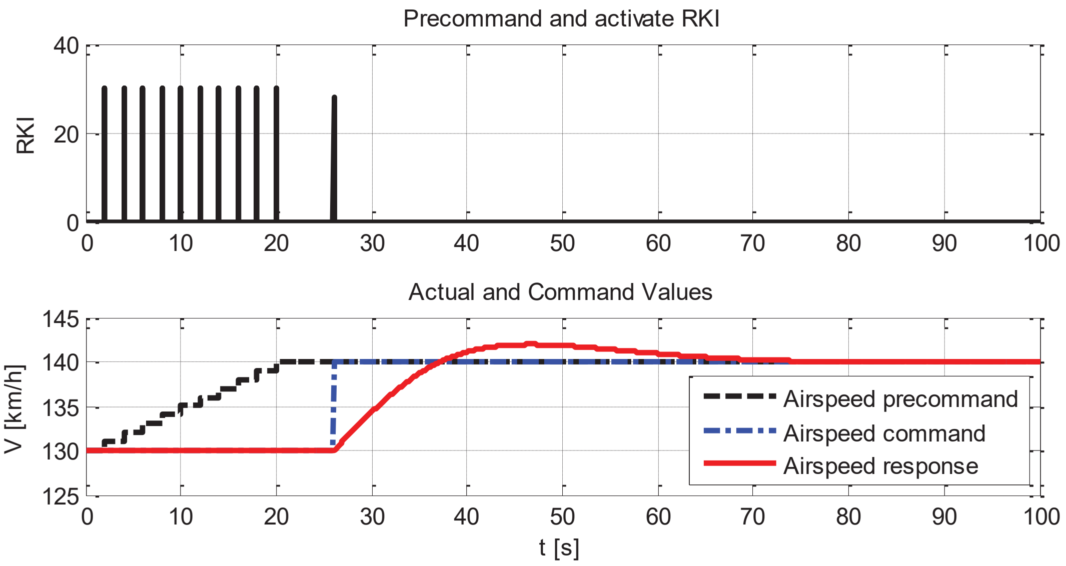


FIGURE 20. An example verification for actual and command for airspeed

For the first test, RKI for precommand value is given several times then RKI for activate is sent. An example for verification result is given in FIGURE 20. The airspeed knob is turned clockwise for 10 steps from 0 to 20 seconds. Each steps will produce RKI of 30 which means increment of 1 km/h and 10 steps result to 10 km/h. From FIGURE 20, it shows that the airspeed precommand input increases from reference of 130 km/h to 140 km/h. This verifies the read function is able to identify that airspeed knob is turned clockwise for 10 steps and to get airspeed reference of 130 km/h from previous steady flight condition. The precommand changes is displayed correctly on command value indicator. In addition, the precommand input does not violate limit function hence the input is kept displayed its designated value. Up until 20 seconds the airspeed response still maintains its reference at 130 km/h which is displayed in on actual value indicator. This verifies that precommand input has not activated and only is displayed in command value. At 25 seconds, the airspeed knob is pressed and this generates RKI of 28 which is plotted in Fig. 20. Subsequently, the activate function detects the RKI for airspeed command activate, sets the final airspeed precommand input of 140 km/h to be airspeed command of 140 km/h, and transmits to FCLs. The result shows that the airspeed responses by following the given command. The airspeed response is depicted on actual value indicator. From the first test result, it can be verified that requirements for changing input values, value change increment, read function, limit function, activate function, actual value, and command value are complied.

The second test is to verify vertical mode functionality. For this test, vertical mode is selected at default value. Then, RKI for altitude precommand and activate are given. Subsequently, RKI for vertical speed precommand and activate are inputted. Later RKI for vertical mode is excited. An example of verification result is given in FIGURE 21. The altitude fine knob is turned clockwise for 5 steps from 0 to 10 seconds. Each steps will produce RKI of 40 which means increment of 10 m and 5 steps result in 50 m. Therefore, the altitude goes from 1000 m to 1050 m. At 13 seconds, RKI of 26 for altitude activate is sent. Later, RKI of 41 for vertical speed precommand is given twice. This means the vertical speed knob is turned clockwise for 2 steps from 20 to 22 seconds which results vertical speed is given precommand from 0 to -1 meter per seconds. At 30 seconds, RKI of 25 for vertical speed activate is sent. Then, RKI of 14 for vertical mode is given at 60 seconds. The second row of FIGURE 21 shows vertical mode select which is displayed on vertical mode value indicator. This means RKI of 14 sent at 60 seconds is verified from vertical mode value. The third row of FIGURE 21 shows altitude precommand from 0 to 10 seconds which is displayed on altitude command value indicator. This verifies the first ten seconds RKI input shown in the first row of this figure. Then, at 13 seconds, altitude command is activated and altitude responses following the altitude command. The altitude response is displayed on altitude actual value indicator. The response of altitude complies selected vertical mode of 1 which is activated until 60 seconds. This verifies that the current vertical mode value only activates altitude control. It is also supported from the given vertical speed precommand at 20 seconds and vertical speed activate at 30 seconds. Those vertical speed control input does not affect the altitude response because the vertical mode is still at altitude mode. Only after vertical mode selected to value of 2 or vertical speed mode, the vertical speed control is active and it can be seen that the vertical speed response follows the vertical speed command and also altitude response no longer holds the current value but drops as consequence of negative vertical speed command. In conclusion, the verification test shows compliances especially toward requirement of vertical mode. In addition, other requirements such as changing input values, value changes increment, read function, limit function, activate function, actual value, command value, and vertical mode value are complied.

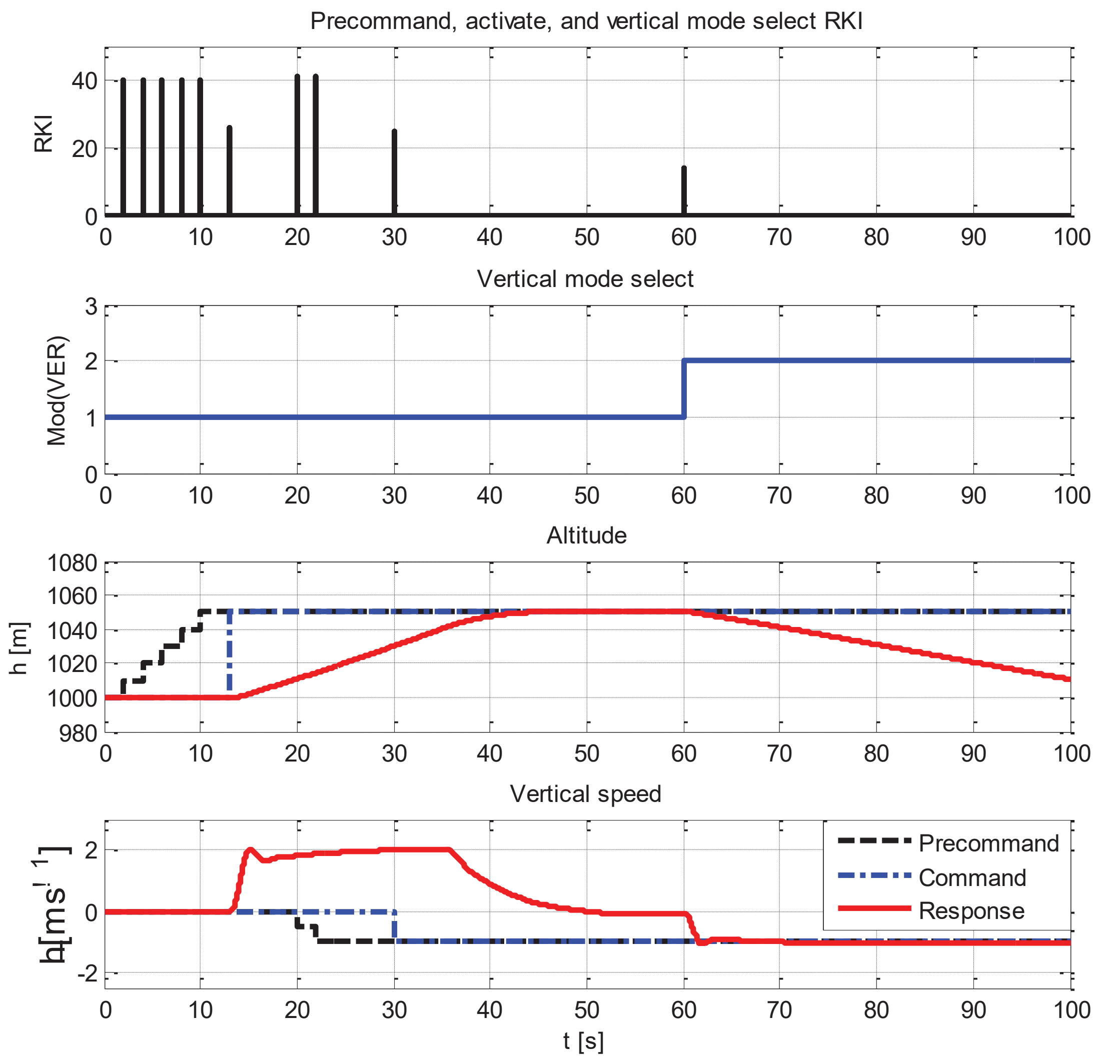


FIGURE 21. An example verification for vertical mode of altitude and vertical speed

CONCLUSIONS

The development of FCPI in MIL level has been accomplished which are composed from two development phases i.e. FCPI concept and requirements definition and FCPI design. The FCPI concept is derived from EFCS phase concept comprised EFCS and FCP engage – disengage sequence and FCP operational concept. The FCPI requirements are written with respect to FCPI components, FCPI parts, FCPI items, button and knobs selection, changing input values, value changes increment, vertical mode, read function, limit function, activate function, actual value, command value, vertical mode value, unit converter, and ASCII converter. FCPI design is shown from block diagram of FCPIC and FCPID. In addition, the equations for read function, limit function, and activate function for airspeed, heading, altitude, vertical speed, and vertical mode have been written.

The verification tests have been defined and performed. The verification shows that FCPI requirements are satisfactorily complied. Verification on architecture such as implementation of FCPIC and FCPID, FCPIC and FCPID parts, and availability of read function, limit function, and activate function has been shown. Verification examples for precommand, activate, and vertical mode have been thoroughly discussed. The examples verifies requirements such as button and knobs selection, changing input values, value changes increment, vertical mode, read function, limit function, activate function, actual value, command value, and vertical mode value.

REFERENCES

1. Bahri, S. Development of Flight Control Laws for The Basic Electronic Flight Control Systems of The LSA-02 Technology Demonstrator Aircraft. (Institut Teknologi Bandung, 2018).
2. Bahri, S. Longitudinal flight control laws for high aspect ratio light utility aircraft. in *6th International Seminar of Aerospace Science and Technology -ISAST 2018* (IOP Conf. Series: Journal of Physics: Conf. Series, 2018). doi:10.1088/1742-6596/1130/1/012026
3. Suseno, P. A. P. & Bahri, S. Experimental flight control function for electronic flight control system of high aspect ratio light utility aircraft. in *7th International Seminar on Aerospace Science and Technology – ISAST 2019* (AIP Conference Proceedings, 2019). doi:https://doi.org/10.1063/5.0002315
4. Bahri, S. & Sasongko, R. A. Development of Nonlinear Flight Mechanical Model of High Aspect Ratio Light Utility Aircraft Development of Nonlinear Flight Mechanical Model of High Aspect Ratio Light Utility Aircraft. in *5th International Seminar of Aerospace Science and Technology* (2018).
5. Lamp, M. & Luckner, R. Flight Control Law Development for the Automatic Flight Control System LAPAZ. in *Conference in Guidance, Navigation and Control in Aerospace* (2011).
6. Wirawan, A. Development and Test of The Software for The Flight Control Panel of LSA-02 Technology Demonstrator Aircraft. (Institut Teknologi Bandung, 2018).
7. Dalldorff, L., Luckner, R. & Reichel, R. A Full-Authority Automatic Flight Control System for the Civil Airborne Utility Aircraft S15 – LAPAZ. 887–906 (2013).
8. Wirawan, A. & Indriyanto, T. Design of Flight Control Panel Layout using Graphical User Interface in MATLAB. in *5th International Seminar of Aerospace Science and Technology* (2018).
9. *Certification Considerations For Highly Integrated or Complex Aircraft Systems*. (PA: SAE ARP 4754, 1996).
10. Forsberg, K. & Mooz, H. 7.17. System Engineering for Faster, Cheaper, Better. *INCOSE Int. Symp.* **8**, 917–927 (1998).
11. Forsberg, K. & Mooz, H. The Relationship of System Engineering to the Project Cycle. in *The 12th INTERNET World Congress on Project Management* (1994).
12. Wirawan, A. & Jayanti, E. B. Desain Awal Flight Control Panel dari Pesawat LSA-02. in *Seminar Nasional Instrumentasi, Kontrol dan Otomasi (SNIKO) 2018* (2018).