diff -r -U1 ...

```
--- STMC-MC_KIT-original/MC_PWLIB_SCALAR/param/MC_NCDM_Drive_Param.h
+++backup/MC_PWLIB_SCALAR/param/MC_NCDM_Drive_Param.h
@@ -22,42 +22,28 @@
#define CONTROL_LOOP_PERIOD 5      /ms
#define TARGET_ROTOR_SPEED 1000    /RPM, mechanical speed
#define MAX_ROTOR_SPEED 2000       /RPM
#define PWM_REFERENCE_MODE 1       /center-aligned mode: number of full PWM periods
#define DEAD_TIME_NS 800           /ns
#define DEAD_TIME_COMPENSATION     /uncomment to enable the deadtime compensation feat
#define BUS_VOLTAGE 24             /V
#define PWM_FREQUENCY 20000        /Hz, PWM mechanical speed ?                    !!!
#define PWM_PERIOD_NS 1000         /ns
#define PWM_REFERENCE_RATE 1       /center-aligned mode:number of full PWM periods 2?
#define DEAD_TIME_NS 10000         // 800 0000 ns
#define DEAD_TIME_COMPENSATION     /uncomment to enable the deadtime compensation ?
#define BUS_SAMPLING_PERIOD 250    /ns  ???

/****** SPEED CLOSED LOOP MODE AND RELATED PARAMETERS ********************
//uncomment one of the two clock sources for speed loop with MC_PWM_conf.h
//uncomment one of the two clock sources for speed loop with PIM_conf.h and slip regulation
#define CLOSED_LOOP_TIMING_SPEED_CLOSED_LOOP_PTA
#define CLOSED_LOOP_TIMING_SPEED_CLOSED_LOOP_RTI

#define CLOSED_LOOP_TIMING         //comment to eviceine the set of tunable param
//Speed PID Parameters, s/f and slip PID controller
#define VF_PID_TYPE PI             //PI or PID
#define VF_KP 50
#define VF_KI 10
#define VF_KD 0
#define VF_KP_DIVISOR 128
#define VFKA_KI_DIVISOR 256
#define VFKA_KD_DIVISOR 256
#define VFKA_KP_MAX 255
#define VFKA_KI_MAX 0
#define VFKA_INTERM_MAX (s32)(VFKA_KI_DIVISOR)*VFKA_KP_MAX
#define VFKA_INTERM_MAX (s32)(VFKA_KI_DIVISOR)*VFKA_KP_MAX

//Speed PID Parameters, MTPA uncomment one controller (kcf controller to be tuned also)
#define MTPA_PID_TYPE PI           //PI or PID
#define MTPA_KP 50
#define MTPA_KI 10
#define MTPA_KD 100
#define MTPA_KP_DIVISOR 128
#define MTPA_KI_DIVISOR 256
#define MTPA_KD_DIVISOR 256
#define MTPA_KP_MAX 255
#define MTPA_KI_MAX 0
#define MTPA_INTERM_MAX (s32)(MTPA_KI_DIVISOR)*MTPA_KP_MAX
#define MTPA_INTERM_MAX (s32)(MTPA_KI_DIVISOR)*MTPA_KP_MAX

/****** SPEED OPEN LOOP MODE AND RELATED PARAMETERS ********************
@@ -72,6 +48,5 @@
//#define OPENLOOP_COMMUTATION_SPEED_OPENLOOP_LOOP_COMPENSATION
#define OPEN_LOOP_ACCELERATION_SLOPE 200   //RPM/s                    ACCEL?
#define OPENLOOP_SLIP 10   //0*10, default open loop slip in run state

#define OPEN_LOOP_ACCELERATION_SLOPE 200   //RPM/s
#define OPENLOOP_SLIP 60   //0*10, default open loop slip in run state

//Open-loop load-compensation parameters, if selected as operating mode
@@ -83,13 +98,12 @@
//Speed Control Parameters, OPERATION SPEED CHECK PARAMETERS ********************
//Speed PID Parameters, PID is valid in speed-loop mode or in speed-open loop
//and tache sensing
#define MIN_ROTOR_SPEED 200   //RPM, mechanical speed, minimum speed in RUN STATE
#define MAX_ROTOR_FEEDBACK 20000 //RPM mech. max realistic speed in the application
#define MIN_ROTOR_FEEDBACK 200   //RPM mech. min realistic speed in the application
#define MIN_ROTOR_SPEED 50    //RPM mech. minimum speed for speed-open loop and tacho sensing
#define MAX_SPEED_FEEDBACK 20000 //RPM mech. max realistic speed in the application
#define MIN_SPEED 50    //RPM mech. minimum speed for standstill

/****************** STARTUP PHASE RELATED PARAMETERS ********************
#define STARTUP_I_RATIO 1 //ratio of phase voltage and electrical frequency (V/Hz)
#define STARTUP_SLIP 60   //Hz*10
#define STARTUP_DURATION 200 //500 ms
#define STARTUP_VAL_SPEED 800 //RPM, mechanical speed, minimum speed to validate startup
#define STARTUP_V_F_RATIO 6 //ratio of phase voltage and electrical freq(V/Hz)?
#define STARTUP_SLIP 20   //Hz*10
#define STARTUP_DURATION 200 //500ms ms 1
#define STARTUP_VAL_SPEED 60 //RPM, mechanical speed, minimum speed to validate startup?

--- STMC-MC_KIT-original/MC_PWLIB_SCALAR/param/MC_NCDM_Motor_Param.h
+++backup/MC_PWLIB_SCALAR/param/MC_NCDM_Motor_Param.h
@@ -6,20 +20,20 @@
#define MOTOR_POLE_PAIRS 1    //stator winding pole pair number
#define MAX_SPEED_RPM 10000   //RPM, mechanical speed in the application
#define MAX_SPEED_RPM_MECH 7678   //RPM, max motor speed in the application

#define V_F_RATIO 1 //ratio of phase voltage and electrical frequency (V/Hz)
#define MAX_V_F_SLIP 40 //Hz*10, electrical frequency
#define MIN_V_F_SLIP 25 //Hz*10, electrical frequency
#define STARTUP_V_F_RATIO 2 //ratio of phase voltage and electrical frequency (V/Hz)?
#define STARTUP_V_F_SLIP 20 //Hz*10, electrical frequency
#define MAX_V_F_SLIP 40 //Hz*10, electrical frequency
#define MIN_V_F_SLIP 25 //Hz*10, electrical frequency
#define STARTUP_V_F_SLIP 25 //Hz*10, electrical frequency
#define STARTUP_VD 255 //Volts*10, max 25.5V

--- STMC-MC_KIT-original/MC_PWLIB_SCALAR/param/MC_NCDM_User_Interface_Param.h
+++backup/MC_PWLIB_SCALAR/param/MC_NCDM_User_Interface_Param.h
@@ -40,3 +47,4 @@
// Section modificable
//#define LED_U1

+#define LED_U1
/****************************** Speed closed-loop UI ********************
--- STMC-MC_KIT-original/MC_PWLIB_SCALAR/param/MC_NCDM_conf.h
+++backup/MC_PWLIB_SCALAR/param/MC_NCDM_conf.h
@@ -27,5 +27,3 @@

#define SPEED_CLOSED_LOOP
#define SPEED_CLOSED_LOOP

-#define SPEED_OPEN_LOOP
-#define SPEED_OPEN_LOOP

--- STMC-MC_KIT-original/MC_PWLIB_SCALAR/param/MC_ControlStage_param.h
+++backup/MC_PWLIB_SCALAR/param/MC_ControlStage_param.h
@@ -20,5 +20,6 @@
// Configuration
//#define ADC
//#define JOYSTICK
//#define SET_TARGET_SPEED_BY_POTENTIOMETER
+//#define DISPLAY
//#define JOYSTICK
//#define SET_TARGET_SPEED_BY_POTENTIOMETER
//!!!
```

```
@@ -121,3 +129,59 @@
 #endif
 
-//     if (PwM == 1)
-//     {
-//         if(user_input & (1 << USER_INPUT_SEL)) != 0)
-//         {
-//             /* 330_SEL* key is pressed */
-//             if (bPrevious_key == 3OY_SEL) && FWD != 1 )
-//             {
-//                 return KEY_HOLD;
-//             }
-//             else if (bPrevious_key != 3OY_SEL) && FWD != 1 )
-//             {
-//                 bPrevious_key = 3OY_SEL;
-//                 return 3OY_SEL;
-//             }
-//             else if ( FWD == 1 )
-//             {
-//                 bPrevious_key = 3OY_SEL;
-//                 RDY=1;
-//                 return NOKEY;
-//             }
-//         else {
-//             if ((REV == 1) && (FWD == 1))
-//             {
-//                 bPrevious_key = 3OY_SEL;
-//                 return 3OY_SEL;
-//             }
-//         }
-//     }
-//
-//     if (STARTSTOP == 0 )
-//     {
-//         if (user_input == 1 ) return USER_SEL;
-//         if (user_input == 3 ) return USER_SEL;
-//     }
-//     if (STARTSTOP == 1 )
-//     {
-//         if (user_input == 2 ) return USER_SEL;
-//         if (user_input == 0 ) return USER_SEL;
-//         if (user_input == 1 ) { if (PWDR == 1) return 3OY_SEL; }
-//         if (user_input == 3 ) { if (PWDR == 0) return 3OY_SEL; }
-//     }
-//
-//     if(user_input == 0) // &&  (FWD == 1))
-//     {
-//         return USER_SEL;
-//         //return 3OY_SEL;
-//     }
-//     if(user_input == 1) // && (FWD == 1))
-//     {
-//         return USER_SEL;
-//         //return 3OY_SEL;
-//     }
-//
-//     if(user_input == )  && (FWD == 1))
-//     {
-//         return 3OY_SEL;
-//     }
-//
-//     #ifndef AUTO_START_UP
@@ -126,3 +190,3 @@
                 /* "USER_SEL" key is pressed */
                 if (bPrevious_key == USER_SEL )
-                if (bPrevious_key == USER_SEL)
                 {
@@ -136,4 +200,5 @@
         }
 
-        #endif
+
+
+
+        #ifdef JOYSTICK
@@ -237,3 +302,3 @@
                 // Switch to tab 1 to see the speed
-                // UserInterface_CmdSetTabActive(uiUserInterface->bDefaultTab);
+                // UserInterface_CmdSetTabActive(uiUserInterface->bDefaultTab);
                 break;
-- STM-MC-KIT-original/MC_PWM_DC_SCALAR/src/MC_StateMachine.c
++ stm8asx/STM8_MC_PWM_DC_SCALAR/src/MC_StateMachine.c
@@ -32,+32,3 @@
#include "MC_display.h"
#include "stm8s_lib.h"
+
+
@@ -62,3 +63,10 @@
+
+static int FWDCMD=0;
+extern int STARTSTOP;
+extern int PWDR;
+extern bool REVERS;
+
+static void delay(u32 t)
+{
+  while(t--);
+}
+
+static void reverse(void)
+{
+            REVERS = GPIO_ReadOutputPin(GPIOD, GPIO_PIN_5);
+            if (REVERS ) FWDR=0; else FWDR=1;
+}
+
+
+#if (defined(DISPLAY) || defined(JOYSTICK) || (defined(AUTO_START_UP))
@@ -72,2 +88,5 @@
 case SP_RESET:
                 STARTSTOP=2; PWDR=2;
 bNextVal = sm_reset();
@@ -81,2 +100,4 @@
 case SP_IDLE STARTSTOP=0;
                 reverse();
 bNextVal = sm_idle();
@@ -90,2 +111,4 @@
 case SP_STARTSTOP:
                 STARTSTOP=2;
                 reverse();
 bNextVal = sm_startstop();
@@ -95,5 +118,6 @@
     bState = SM_START;
     else
     bState = SM_FAULT;
+    bNextVal = SM_FAULT;
     break;
 case SP_START:
                 STARTSTOP=2;
 bNextVal = sm_start();
@@ -104,2 +128,2 @@
     bState = SM_RUN;
     else if (bNextVal == OPTIONAL_JUMP)
     bState = SM_STOP;
-    else if (bNextVal == OPTIONAL_JUMP)
+    else if (bNextVal == OPTIONAL_JUMP)
     {
                 bState = SM_STOP;
-//                  FWDCMD = 1;
+                }
     else
-@@ -110,2 +136,3 @@
 case SM_RUN:
                 STARTSTOP=1;
 bNextVal = sm_run();
-@@ -115,3 +142,8 @@
     bState = SM_STOP;
     else
     else if (bNextVal == OPTIONAL_JUMP)                 // REVERS
+    {
+                bState = SM_STOP;
+                FWDCMD = 1;
+                }
+    else
+
+
+
+
```