Chapter 7
SOFTWARE IMPLEMENTATION

7.1 Introduction

This chapter describes the implementation of microcontroller software to control the quasi resonant converter. It has to take care of load tuning, over current protection and mod-bus communication. It also describes the implementation of Embedded software for implementation of GUI, a discrete Proportional-Integral-Derivative (PID) controller and autotuning.

There are two parts of the complete software.

1. Low-level (Assembly Language) software for Micro-controller Board.
2. High-level (C Language) software for ARM-7 Board.

7.2 Low-level software for Micro-controller Board

The micro-controller board has to perform following tasks.

1. Read Temperature of crucible through thermocouple using Sigma-Delta ADC and do linearization and cold junction compensation.
2. Reading of RMS value of DC-Link voltage & current using sigma-delta ADC and finding DC-Link power.
3. Sending calibrated DC-Link power & linearized temperature to ARM-7 board through MOD-BUS communication.
4. Taking commands of frequency & frequency bursts from ARM-7 board.
5. Generating gate pulses for IGBT using these commands & to automatically tune the on duty cycle of gate pulses according to changes in load inductance.
6. To switch off IGBT gate firing in case of over current in tank circuit, DC-Link over current, DC-Link under voltage, over temperature (sensed through thermostat), driver card fault, etc.

To implement all above tasks the whole software is divided into 5 parts.

7.2.1 Main program

Following is the listing of Main program which initializes the SFR’s to set the modes of timers & serial port and enables interrupts. It checks calibration data stored in Eeprom and validates them & informs ARM-7 board for any error. After which it enables timer & external interrupts as well as serial port interrupt. And finally remains in a loop where it checks for any error & informs ARM-7 board as per error generated.
Chapter 7

SOFTWARE IMPLEMENTATION

MAIN:

MOV SP,#BDH
MOV P0,#11111101B ;KEEP SCLK (CS5460) LOW
MOV P1,#00111111B ;KEEP ENABLE PIN (LCD) LOW, 485 IN RECV. MODE
MOV P2,#FFH
MOV AUXR,#00010001B ;XRAM SIZE 1792 BYTES, INT-XRAM SELECTED
MOV CKCON0,#7FH ;X2 SPEED (3.072 MIPS)
MOV CKCON1,#01H
MOV CKRL,#FFH
MOV PCON,#00H ;RESET POWER-OFF FLAG
MOV AUXR1,#00H ;BOOT ROM DISABLED & DPTR0 SELECTED
MOV WDTPRG,#05H ;TIMEOUT -> 170MS FOR X2 MODE
MOV IPH1,#00000000B
MOV IPL1,#00000000B ;SPI & KBD INTERRUPTS ARE NOT USED
MOV IPH0,#00001101B
MOV IPL0,#00001101B ;SPI & KBD INTERRUPTS ARE NOT USED
MOV TCON,#00000101B ;INT0,1 FALLING EDGE
MOV TMOD,#00010001B ;TIMER 1,0 IN TIMER MODE
MOV TH1,#FFH
MOV TL1,#66H ;COUNT FOR ON CYCLE TIME = 100uS
MOV TH0,#FAH
MOV TL0,#07H ;COUNT FOR 1MS
MOV TR0
MOV TH1,#FFH
MOV TL1,#66H ;COUNT FOR ON CYCLE TIME = 100uS
MOV TR1
MOV ET1
MOV T2CON,#00000000B ;T2 -> 16BIT AUTORELOAD MODE
MOV RCAP2H,#10H
MOV TH2,#10H
MOV RCAP2L,#00H
MOV TL2,#00H ;COUNT FOR 40MS
MOV TR2
ORL PCON,#00000000B ;SMOD1 = 0
CLR TI
MOV BRL,#246 ;9600 BAUD (X2-SPEED)
MOV BDRCON,#00011100B ;START BRG, USE BRG FOR TX,RX BOTH
MOV SCON,#60H ;SERIAL MODE-1, WITH SM2=1

MOV 0FH,#FDH
MOV 0EH,#04H ;COUNT FOR OFF CYCLE TIME FOR 1.67KHZ
MOV 31H,#250 ;CS5460-1 TIMEOUT = 10 SEC
MOV 35H,#250 ;CS5460-2 TIMEOUT = 10 SEC
MOV 38H,#30
MOV 39H,#00H
MOV 3AH,#00H ;INIT ROOM TEMP. AS 30 DEG
MOV 3BH,#01H ;RESET KEY-DEBOUNCE COUNTER
MOV 3FH,#00H ;INIT PRESENT POWER O/P = 0KW
MOV 40H,#00H ;MAKE PRESENT O/P FREQ. = 1.67KHZ
MOV 59H,#10 ;COUNT FOR 1 SEC
MOV 5AH,#03H  ;COUNT FOR 0.12 SEC
MOV 5BH,#25  ;COUNT FOR 1 SEC
MOV 5CH,#00H  ;RESET ERROR BYTE
MOV 5EH,#00H  ;RESET OPERATION MODE BYTE TO POWER OFF MODE
MOV 67H,#A8H  ;INIT RECV_BUF POINTER
MOV 6AH,#10  ;TO GIVE DEBOUNCE TO CJ-ERR (10 SECONDS)
MOV 6BH,#06H  ;TO DELAY XMISSION ATLEAST BY 6MS
MOV 70H,#10  ;COUNT FOR 10MS
MOV 71H,#03H  ;RESET UNDER VOLTAGE FAULT CHECK TRIAL CNTR
LCALL DLY_20MS
:RESET CS5460-1,2 SERIAL PORT
CLR P0.1
NOP
NOP
NOP
NOP
NOP
APPLICATION H/W RESET
MOV 30H,#00H  ;TO CALL INIT5460-1 FOR 1ST TIME
MOV 34H,#00H  ;TO CALL INIT5460-2 FOR 1ST TIME
MOV WDTRST,#1EH
MOV WDTRST,#E1H  ;RESET WATCH-DOG TIMER TO ENABLE IT
SETB ET2
SETB EA
CLR P2.1  ;REMOVE RESET SIGNAL OF IGBT-DRIVER
LCALL RDY_EE  ;WAIT TILL EEPROM IS READY
JCB ERR1
LCALL BYTE_FIL
JCB ERR1
LCALL VERFY
JNC OK24512
ERR1:
LCALL ERR2
OK24512:
MOV 32H,#00H  ;RELOAD COUNT FOR 10SEC
SETB 05H  ;START KBD TIMER (IF OFF)
LCALL VALID1  ;CHECK FOR VALIDITY OF DATA IN EEPROM
JNC A2
MOV 5CH,#03H  ;FLAG THAT DATA IS INVALID
LCALL W_KBD  ;WAIT FOR A KEY STROKE
LCALL DEFLT1  ;LOAD DEFAULT VALUES IN EEPROM
LCALL DLY_0.5
MOV 5CH,#00H  ;RESET ERROR BYTE
A2:
LCALL VALID11  ;CHECK IMP. DATA IN EEPROM
JNC AA2
MOV 5CH,#03H  ;FLAG THAT DATA IS INVALID
LCALL W_KBD  ;WAIT FOR A KEY STROKE
SOFTWARE IMPLEMENTATION

```assembly
LCALL DEFLT11 ;LOAD DEFAULT VALUES IN EEPROM
LCALL DLY_0.5
MOV 5CH,#00H ;RESET ERROR BYTE
AA2:
LCALL VALID2 ;CHECK FOR VALIDITY OF DATA IN EEPROM
JNC A4
MOV 5CH,#03H ;FLAG THAT DATA IS INVALID
LCALL W_KBD ;WAIT FOR A KEY STROKE
LCALL DEFLT2 ;LOAD DEFAULT VALUES IN DS12887
LCALL DLY_0.5
MOV 5CH,#00H ;RESET ERROR BYTE
A4:
LCALL LD_VARS ;LOAD VARIABLES FROM EEPROM INTO INT.XRAM
CLR 0BH ;RESET CS5460
-1 FAULT BIT TO START ACCESSING
CLR 0CH ;RESET CS5460
-2 FAULT BIT TO START ACCESSING
; SETB ET0
CLR RI ;DISCARD RI (DUE TO ANY REFLECTIONS)
SETB REN
SETB ES ;ENABLE RECEPTION (FOR MODBUS COMMUNICATION)
SETB 10H ;ALLOW ACCESS TO I2C LINE IN INT.
LCALL DELAY_1S
CLR P2.7 ;SW.ON CONTACTOR TO ACTIVATE POWER
LCALL DELAY_1S ;ALLOW CS5460 TO START
SETB 41H ;ACTIVATE DC-LINK UNDER-VOLT FAULT CHECKING
ST0:
JBC 42H,UNDERV1 ;CHECK UNDER-VOLTAGE FAULT
JB 59H,EMG1 ;CHECK EMERGENCY SWITCH
JBC 53H,OCERR1 ;CHECK OVER-CURRENT FAULT
JBC 55H,IEXCEED1 ;CHECK I/P-CURRENT EXCEED FAULT
JB 4EH,DRVER1
JB 0BH,ER54601
JB 0CH,ER54601
JBC 1EH,ERR22
JBC 02H,CRCERR
JBC 06H,CJTERR
SJMP ST0

Table-7.1 Main program listing (Micro-controller board)

7.2.2 Timer-0 Interrupt at 1ms
In this subroutine the MOD-BUS timings are taken care off. If Melter power is on then it generates IGBT on/off commands as per required frequency bursts.

T0_OVF:
MOV TH0,#FAH ;COUNT FOR 1MS
MOV TL0,#07H ;(LESS BY 7 CYCLES TO GET EXACT 1MS TIME)
JNB 2AH,I60
```
DJNZ  6BH,160
MOV   6BH,#06H ;TO DELAY XMISSION ATLEAST BY 6MS
CLR   2AH
SETB  TI ;START TRANSMISSION

I60:
  JNB   2BH,161
  DJNZ  6CH,161
  MOV   6CH,#50 ;COUNT FOR 50MS
  MOV   67H,#A8H ;RE-INIT RECV_BUF POINTER FOR RE-SYNCHRONISM
  CLR   2BH

I61:
  JNB   2CH,162
  DJNZ  6DH,162
  CLR   2CH
  CLR   RI ;DISCARD RI (DUE TO ANY REFLECTIONS)
  SETB  REN
  MOV   67H,#A8H ;RE-INIT RECV_BUF POINTER FOR RE-SYNCHRONISM

I62:
  DJNZ  70H,RET30
  MOV   70H,#10 ;RELOAD COUNT FOR 10MS
                   ;10MS
  PUSH  PSW
  PUSH  A
  CLR   RS1
  SETB  RS0 ;R.B - 1
  JNB   52H,P_OFF ;CHECK POWER ON/OFF BIT
                   ;SWITCH IGBT ON/OFF BIT ACCORDING TO ITS ON/OFF TIMER
  MOV   R0,#B1H
  MOV   A,@R0
  DEC   R0
  DEC   @R0
  CJNE  @R0,#00H,164 ;CHECK FOR CT
  MOV   @R0,A ;RESET CYCLE TIME TIMER (HEAT)
  MOV   R0,#BAH
  MOV   A,@R0
  INC   R0
  INC   R0
  ORL   A,@R0
  JNZ    ON1
  INC   R0
  MOV   A,@R0
  DEC   R0
  MOV   @R0,A ;RELOAD PRECALCULATED OFF CYCLE COUNT
  DEC   R0
  MOV   A,@R0
  DEC   R0
  MOV   @R0,A ;RELOAD PRECALCULATED ON CYCLE COUNT
ON1:
    MOV   R0,#BAH
    MOV   A,@R0
    JZ    OFF1    ;CHECK FOR ON CYCLE COUNT
    DEC   @R0
    SETB  50H     ;START SWITCHING IGBT
    SJMP   I64
OFF1:
    INC   R0
    INC   R0
    MOV   A,@R0
    JZ    I64     ;CHECK FOR OFF CYCLE COUNT
    DEC   @R0
P_OFF:
    CLR   50H     ;STOP SWITCHING IGBT
I64:
    POP    A
    POP    PSW
    RET30:
    RETI

Table 7.2 Timer-0 Interrupt service subroutine listing

7.2.3 Timer-1 Interrupt for maximum on-cycle time
This subroutine takes care of maximum on-cycle time and generates firing signals for IGBT.

T1_OVF:
    JBC   51H,I104
    MOV   TH1,#FFH
    MOV   TL1,#66H   ;COUNT FOR ON CYCLE TIME = 100uS
    JNB   50H,RET12  ;CHECK IGBT ON/OFF COMMAND
    SETB  51H        ;FLAG THAT GATE IS ON
    CLR   P2.2      ;SW.ON THE GATE
    CLR   IE0       ;DISCARD PREVIOUS EDGES
    SETB  EX0       ;ENABLE INT0
RET12:
    RETI
I104:
    SETB   P2.2    ;SW.OFF GATE
    JBC   EX0,I105
    RETI
I105:
    PUSH   PSW
    CLR    RS1
    SETB   RS0     ;R.B - 1
    CJNE   R7,#FFH,I107
    CJNE   R6,#B5H,I107
    I107:
Table-7.3 Timer-1 Interrupt service subroutine listing

7.2.4 Timer-2 Interrupt at 40ms

It reads DC-Link voltage and current from CS5460 and then calculates DC-Link power. Then it reads
temperature and does linearization and cold junction compensation. In the later portion of this
subroutine it applies accelerating/decelerating slope to the frequency of IGBT gate pulses as per
start/stop command.

T2_OVF:
CLR TF2
PUSH PSW
PUSH A
PUSH B
PUSH DPH
PUSH DPL
ORL PSW,#00011000B ;R.B.-3
MOV WDTRST,#1EH
MOV WDTRST,#E1H
;READING CS5460-1 STARTS...
JB 0BH,I32 ;IF 5460-1 IS FAULTY THEN SKIP READING
CLR P0.4 ;SELECT 5460-1
MOV A,30H
JNZ NORST1 ;IF RESET IS NOT IN PROGRESS
MOV R2,#00H ;CURRENT GAIN = 10
LCALL INIT5460
MOV 36H,#06H ;DISCARD 5 SAMPLES AFTER INITIALIZATION
MOV 30H,#12 ;REINIT TRIAL COUNTER
LJMP I31
NORST1:
MOV A,#00011110B ;READ STATUS REG. COMMAND
LCALL W_5460
LCALL R_5460 ;DUMMY READ OF STATUS REGISTER
MOV R7,A
LCALL R_5460 ;DUMMY READ A BYTE FROM CS5460
LCALL R_5460 ;DUMMY READ A BYTE FROM CS5460
MOV A,R7
JNB ACC.7,130 ;IF DRDY IS NOT ACTIVE
MOV A,#00011000B ;READ V-RMS REG. COMMAND
LCALL W_5460
LCALL R_5460 ;READ A BYTE FROM CS5460
MOV 79H,A
LCALL R_5460 ;READ A BYTE FROM CS5460
MOV 78H,A ;SAVE 16BIT V-RMS DATA
LCALL R_5460 ;DUMMY READ A BYTE FROM CS5460
MOV A,#00010110B ;READ I-RMS REG. COMMAND
LCALL W_5460
LCALL R_5460 ;READ A BYTE FROM CS5460
MOV 77H,A
LCALL R_5460 ;READ A BYTE FROM CS5460
MOV 76H,A ;SAVE 16BIT I-RMS DATA
LCALL R_5460 ;DUMMY READ A BYTE FROM CS5460
MOV A,#01011110B ;WRITE STATUS REG. COMMAND
LCALL W_5460
MOV A,#80H
LCALL W_5460
MOV A,#00H
LCALL W_5460
MOV A,#00H ;RESET DRDY INT.STATUS BIT
LCALL W_5460
SETB P0.4 ;DISABLE 5460-1
MOV 30H,#12 ;REINIT TRIAL COUNTER
MOV 31H,#250 ;RELOAD CS5460 TIMEOUT = 10 SEC
DJNZ 36H,I32 ;DISCARD 5 SAMPLES AFTER INITIALIZATION
MOV 36H,#01H ;NOW DON'T DISCARD ANY SAMPLE
SETB 09H ;FLAG THAT VRMS,IRMS DATA IS AVAILABLE
LCALL CAL_VI
LCALL CH_VRMS ;CHECK V_RMS FOR UNDER VOLTAGE FAULT
LCALL CH_IRMS ;COMPARE I_RMS WITH I/P CURRENT MAX. LIMIT
LCALL CAL_PWR ;CALCULATE POWER FROM V_RMS & I_RMS
II32:
SJMP I32
I30:
DJNZ 31H,I30
MOV 31H,#250 ;CS5460 TIMEOUT = 10 SEC
SETB 0BH ;FLAG THAT CS5460-1 IS FAULTY
II30:
DJNZ 30H,I31 ;CHECK FOR 12 TRIALS (0.48SEC)
;RESET CS5460 SERIAL PORT
CLR P0.1
NOP
NOP
NOP
NOP
NOP
SETB P0.1 ;APPLY H/W RESET
SOFTWARE IMPLEMENTATION

MOV 34H,#00H ;ALSO CALL INIT5460-2
SETB P0.4 ;DISABLE 5460-1
LJMP I10 ;SKIP 5460-2 & ALL ITS CALCULATIONS
I31:
SETB P0.4 ;DISABLE 5460-1
I32:
;READING CS5460-2 STARTS...
JB 0CH,III10 ;IF 5460-2 IS FAULTY THEN SKIP READING
CLR P0.5 ;SELECT 5460-2
MOVA.34H
JNZ NORST2 ;IF RESET IS NOT IN PROGRESS
MOV R2,#01H ;CURRENT GAIN = 50
LCALL INIT5460
MOV 37H,#06H ;DISCARD 5 SAMPLES AFTER INITIALIZATION
MOV 34H,#12 ;REINIT TRIAL COUNTER
LJMP I34
NORST2:
MOV A,#00011110B ;READ STATUS REG. COMMAND
LCALL W_5460
LCALL R_5460 ;DUMMY READ OF STATUS REGISTER
MOV R7,A
LCALL R_5460 ;DUMMY READ A BYTE FROM CS5460
LCALL R_5460 ;DUMMY READ A BYTE FROM CS5460
MOV A,R7
JB ACC.7,I33 ;IF DRDY IS ACTIVE
DJNZ 35H,II34
MOV 35H,#250 ;CS5460 TIMEOUT = 10 SEC
SETB 0CH ;FLAG THAT CS5460-2 IS FAULTY
LCALL RESTMR ;RESET ON/OFF CYCLE TIMERS
LCALL HT_OFF ;SW.OFF MELTER POWER
LCALL STP_TMR ;STOP O/P ON/OFF CHECK TIMER
LCALL SW_OFF ;SW. OFF ALARM RELAYS
II34:
DJNZ 34H,I34 ;CHECK FOR 12 TRIALS (0.48SEC)
;RESET CS5460 SERIAL PORT
CLR P0.1
NOP
NOP
NOP
NOP
NOP
SETB P0.1 ;APPLY H/W RESET
MOV 30H,#00H ;ALSO CALL INIT5460-1
I34:
SETB P0.5 ;DISABLE 5460-2
III10:
LJMP I10 ;SKIP ALL CALCULATIONS
I33:
;0.1SEC
    MOV  A,#00011000B ;READ V-RMS REG. COMMAND
    LCALL  W_5460
    LCALL  R_5460  ;READ A BYTE FROM CS5460
    MOV  R7,A
    LCALL  R_5460  ;READ A BYTE FROM CS5460
    MOV  R6,A  ;SAVE 16BIT C.J.TEMP. DATA
    LCALL  R_5460  ;DUMMY READ A BYTE FROM CS5460
    MOV  A,#00010110B ;READ I-RMS REG. COMMAND
    LCALL  W_5460
    LCALL  R_5460  ;READ A BYTE FROM CS5460
    MOV  R2,A
    LCALL  R_5460  ;READ A BYTE FROM CS5460
    MOV  R5,A
    LCALL  R_5460  ;READ A BYTE FROM CS5460
    MOV  R4,A
    MOV  A,#01011110B ;WRITE STATUS REG. COMMAND
    LCALL  W_5460
    MOV  A,#80H
    LCALL  W_5460
    MOV  A,#00H
    LCALL  W_5460
    MOV  A,#00H  ;RESET DRDY INT.STATUS BIT
    LCALL  W_5460
    SETB  P0.5  ;DISABLE 5460-2
    MOV  34H,#12  ;REINIT TRIAL COUNTER
    MOV  35H,#250  ;RELOAD CS5460 TIMEOUT = 10 SEC
    DJNZ  37H,III10  ;DISCARD 5 SAMPLES AFTER INITIALIZATION
    MOV  37H,#01H  ;NOW DON'T DISCARD ANY SAMPLE
    LCALL  HALF
    LCALL  HALF  ;CONVERT INTO 14BIT FOR COLD-JUNCTION TEMP.
    MOV  7FH,R7
    MOV  7EH,R6  ;SAVE 14BIT C.J.TEMP. DATA
    LCALL  SH_LFT
    MOV  A,#00H
    RLC  A
    MOV  B,A
    LCALL  SH_LFT
    MOV  A,B
    RLC  A  ;CONVERT 24BITS TO 18BITS
    MOV  R0,#80H
    MOV  @R0,1DH
    INC  R0
    MOV  @R0,1AH
    INC  R0
    MOV  @R0,A  ;SAVE 18BITS DATA IN INT. RAM
    SETB  00H  ;FLAG THAT ADC DATA IS AVAILABLE
    LCALL  C_CJTMP  ;CALCULATE COLD JUNCTION TEMP.
SOFTWARE IMPLEMENTATION

LCALL LINEAR ;CALIBRATE & LINEARIZE TEMPERATURE

; ACC. SLOPE OF IGBT FIRING FREQUENCY AS PER SET O/P POWER (0-10KW -> 10SEC)
JB 52H, I92 ; IF POWER IS ON THEN GO FOR ACC/DEC SLOPE
CLR EA
MOV 0FH,#FDH
MOV 0EH,#04H ; COUNT FOR OFF CYCLE TIME FOR 1.67KHZ
SETB EA
MOV 40H,#00H ; MAKE PRESENT O/P FREQ. = 1.67KHZ
SJMP I99

I92:
JNB 54H, I99 ; IF SLOPE IS NOT REQUIRED
MOV A, 42H ; TAKE ONLINE REQD. O/P POWER
JZ DECSL ; IF POWER IS TO BE SWITCHED OFF
DJNZ 5AH, I99
MOV 5AH, #03H ; COUNT FOR 0.12 SEC

; 0.12SEC
CJNE A, 3FH, I93 ; COMPARE WITH PRESENT O/P POWER
SJMP I95

I93:
JC DECSL
MOV A, 40H
CJNE A, #100, I94

I94:
JNC I96
INC 40H
MOV A, 40H
SJMP I96

DECSL:
DJNZ 40H, I95
CLR EX1 ; DISABLE INT1 (TO MONITOR OVER-CURRENT FAULT)
CLR 52H ; SW. OFF THE MELTER POWER

I95:
MOV A, 40H

I96:
ADD A, A
MOV B, A
MOV DPTR, #CNTTBL
MOVC A, @A+DPTR ; AS MAX. PWR IS 100 THIS WILL WORK OK
CLR EA
MOV 0FH, A
INC DPTR
MOV A, B
MOVC A, @A+DPTR
MOV 0EH, A ; UPDATE COUNT FOR OFF CYCLE TIME
SETB EA

I99:
POP DPL
Table-7.4 Timer-2 Interrupt service subroutine listing

7.2.5 Interrupt on external falling edge (generated by Load Tuning Circuit of Figure 6-7)

INTO_ISS:
SETB P2.2 ;SW.OFF GATE
RETI

Table-7.5 External Interrupt service subroutine listing

7.2.6 Serial port Interrupt

Here the serial transmission & reception is implemented using MODBUS protocols. It acts as the slave and responds to the commands received from ARM-7 board. It sends DC-Link power & linearized temperature to ARM-7 and receives the frequency of operation as well as frequency bursts parameters from ARM-7 board.

SRL_ISS:
JBC RI,RECV
CLR TI
DJNZ 69H,M51 ;69H = NO.OF CH. IN XMIT BUF. + 1
SETB 04H ;FLAG THAT COMPLETE TRANSMISSION IS OVER
CLR P1.7 ;CONVERT INTO RECV MODE
MOV 6DH,#04H ;TIME BETWEEN SENDING & RECEIVING = 4MS
SETB 2CH ;ENABLE RECESSION AFTER ATLEAST 4MS
RETI

M51:
PUSH PSW
CLR RS1
SETB RS0 ;R.B - 1
CLR 04H ;TRANSMISSION IS GOING ON
MOV R0,68H
MOV SBUF,@R0
INC 68H
POP PSW
RET2:
RETI

RECV:
JNB REN,RET2 ;AS PER 89C51ED2-ERRATA WORK AROUND
PUSH A
MOV A,SBUF
PUSH    PSW
PUSH    B
PUSH    DPH
PUSH    DPL
CLR     RS1
SETB    RS0    ;R.B - 1
MOV     6CH,#50    ;COUNT FOR 50MS
SETB    2BH    ;START 50MS RECEPTION-TIMEOUT TIMER
MOV     R0,67H
MOV     @R0,A
INC     67H
MOV     A,67H
CJNE    A,#B0H,RET01
MOV     67H,#A8H
LCALL   C_CRC_R    ;CALCULATE CRC FROM RECV.BUFFER(DPH,DPL)
MOV     R0,#AEH
MOV     A,@R0
CJNE    A,DPL,M55
INC     R0
MOV     A,@R0
CJNE    A,DPH,M55
CLR     02H    ;IF NO CRC ERROR THEN RESET COMM. ERROR BIT
SJMP    M56
M55:
SETB    02H    ;IF CRC ERROR THEN SET COMM. ERROR BIT
RET01:
LJMP    RET0
M56:
MOV     DPTR,#00D8H
MOVX    A,@DPTR    ;GET M/C ID. NO.
; MOV     B,A
MOV     B,#01H    ;FIX M/C ID. NO.=01
MOV     R0,#A8H
MOV     A,@R0
JZ      M57    ;IF GLOBLE ADDRESS
CJNE    @R0,#03H,M65    ;CHECK FOR READ HOLDING REGISTER CMD
INC     R0
CJNE    @R0,#00H,ADRER1    ;IF ADDR. DOESNT MATCH THEN SEND EXCPN RESP.
INC     R0
MOV     A,@R0
CJNE    A,#12H,M58
JNC     ADRER1    ;IF ADDR. DOESNT MATCH THEN SEND EXCPN RESP.
ACALL   RDHREG    ;PROCESS READ HOLDING REGISTER COMMAND

M57:
;CHECK & ANALYZE THE COMMAND
MOV     R0,#A9H
CJNE    @R0,#03H,M56    ;CHECK FOR READ HOLDING REGISTER CMD
INC     R0
CJNE    @R0,#00H,ADRER1    ;IF ADDR. DOESNT MATCH THEN SEND EXCPN RESP.
INC     R0
MOV     A,@R0
CJNE    A,#12H,M58
M58:
JNC     ADRER1    ;IF ADDR. DOESNT MATCH THEN SEND EXCPN RESP.
ACALL   RDHREG    ;PROCESS READ HOLDING REGISTER COMMAND
AJMP RET0

M65:
  CJNE  @R0,#06H,M72 ;CHECK FOR PRESET SINGLE REGISTER CMD
  INC   R0
  CJNE  @R0,#00H,ADRER1 ;IF ADDR. DOESNT MATCH THEN SEND EXCPN RESP.
  INC   R0
  MOV   A,@R0
  CJNE  A,#12H,M67

M67:
  JNC    ADRER1          ;IF ADDR. DOESNT MATCH THEN SEND EXCPN RESP.
  ACALL  PRSREG          ;PROCESS PRESET SINGLE REGISTER COMMAND
  SJMP   RET0

ADRER:
  MOV   R0,#A8H
  MOV   A,@R0
  JZ     RET0            ;DONT SEND RESPONSE FOR BROAD CAST MESSAGE
  JNB    04H,\$          ;WAIT TILL PREVIOUS TRANSMISSION IS OVER
  MOV   R0,#99H
  MOV   @R0,B
  MOV   R0,#A9H
  MOV   A,@R0           ;TAKE FUNCTION CODE
  SETB   ACC.7
  MOV   R0,#9AH
  MOV   @R0,A
  INC   R0
  MOV   @R0,#02H        ;ILLEGAL DATA ADDRESS
  SJMP   M79

VALER:
  MOV   R0,#99H
  MOV   @R0,B
  MOV   R0,#A9H
  MOV   A,@R0           ;TAKE FUNCTION CODE
  SETB   ACC.7
  MOV   R0,#9AH
  MOV   @R0,A
  INC   R0
  MOV   @R0,#03H        ;ILLEGAL DATA VALUE

M79:
  MOV   R2,#03H
  LCALL   CALCRC
  INC   R0
  MOV   @R0,DPL
  INC   R0
  MOV   @R0,DPH
  MOV   69H,#06H        ;5 NO.OF CH. IN XMIT BUFFER
  LCALL   XMIT

RET0:
  POP    DPL
Table 7.6 Serial Port Interrupt service subroutine listing

POP DPH
POP B
POP PSW
POP A
RETI
7.3 High-level software for Embedded controller (ARM-7) Board

The Embedded controller (ARM-7) board has to perform following tasks.

1. Interfacing TFT display & touch screen.
2. To implement graphical user interface for displaying & modifying machine parameters.
3. Reading calibrated DC-Link power & linearized temperature from micro-controller board through MOD-BUS communication.
4. Implement PID on temperature & DC-Link power to determine frequency of operation & frequency bursts and send them to micro-controller board.
5. Implement Auto-tune for PID loop to optimize the control.

7.3.1 Graphical User Interface

As the monitoring quantities are Crucible temperature, DC-Link Voltage & DC-Link Current they are displayed as shown in Figure 7.1. From this figure it is also seen that three buttons are kept in the bottom part of GUI. The first START button is used to start the Melter power, the second STOP button is used to stop the Melter power and the third Right Arrow button is used to go to next page to change different parameters.

![Figure 7-1 Monitor Page of GUI](image_url)
By pressing the right arrow the parameter setup page as shown in Figure 7.2 is opened. Here it is possible to set required output power, Alarm set point, Unit of temperature displayed, date & time.

Figure 7-2 Setup Page of GUI
SOFTWARE IMPLEMENTATION

To implement the required tasks the whole software is divided into 5 parts. The program is written in C-Language using Keil-Real-View as shown in Figure 7-3.

As seen from Figure 7.3 the whole project is distributed in different groups.

1. **Startup Code**: Here stack configuration is defined and system as well as peripheral clocks are configured and started. It also initializes the external dynamic memory interface & external static memory interface.
2. **Eeprom**: Contains driver routines for AT24C512 Eeprom which is used to store non-volatile data.
3. **Nor Flash**: Contains driver routines for reading & writing from & to the nor flash which is used to store the still images (Initial title pages).
4. **Touch Screen**: Contains driver routines for interfacing TSC2046 touch screen controller as well as 4-point calibration routines for touch screen.
5. **TFT display**: Contains driver routines that initialize peripheral for TFT display. It also includes all low level graphics functions to create images.
6. **Source Code**: The most important part of the whole project is source code which includes MAIN.C, IRQ1.C & UART.C
7.3.2 Main program (MAIN.C)
Following is the listing of Main program which initializes the ports for input/output configuration. And then sets the modes of timers & serial port and enables interrupts. Finally enters into a loop where it continuously display the Set & Present temperature as well as DC-Link voltage & current. It also checks for touch screen for any key pressed and sends commands to micro-controller card (through MOD-BUS) to start/stop the Melter power.

```c
int main(void)
{
  tU8  ee_buf[128], temp;
  tS32  x, y, z = 0;
  tU32  i, j;
  SCS | 0x01;  // High speed GPIO on Port 0 & 1
  PCONF & 0x00000086;  // Switch off power to unused blocks
  // Set PO.23-PO.30 for Output (TO CHECK INTERRUPT TIMINGS)
  PINSEL1 &= 0xC3FFFFFF;
  PIO0DIR &= 0x60000000;
  // Set FOR P0-PUR-B Output, SET dir & output ON P2.25
  PINSEL5 &= 0xFFF8FFFF;
  PIO2DIR &= 0x1<<25;
  PIO2CLR &= 0x1<<25;  // KEEP USB-POWER OFF
  // Set P0.4-P0.11 for Output ROW0-ROW7
  PINSEL0 &= 0xFF0000FF;
  PIO0DIR &= 0x000000FF;
  PIO0CLR &= 0x000000FF;
  // Set P1.6-P1.13 for Input COL0-COL7
  PINSEL2 &= 0xF00000FF;
  PIO1DIR &= 0x0FFFFFFF;
  // Set P0.19-P0.32, P4.26 for Output DIG0-DIG4
  PINSEL1 &= 0xFFFFC00F;
  PINSEL9 &= 0xFFFFFFFF;
  PIO0DIR &= 0x00780000;
  PIO0SET &= 0x00780000;
  PIO4DIR &= 0x04000000;
  PIO4SET &= 0x04000000;
  // Set P2.10 for Input
  PINSEL4 &= 0x8F0000FF;
  PIO2DIR &= 0x01<<10;
  // Set P1.0-P1.5 for Output SEG0-SEG5
  PINSEL2 &= 0x00000003;
  PIO1DIR &= 0x00000003;
  PIO1SET &= 0x00000003;
  // Set P1.14,P1.15 for Digital Input DI1-DI2
```

Table-7.7 MAIN.C Part-1 listing
SOFTWARE IMPLEMENTATION

Table 7.8 MAIN.C Part-2 listing

```c
/* Enable and setup timer-0 interrupt, start timer */

PCONP |= (1 << 1); /* Enable power to TIM block */
TICK0 = 876000; /* 1ms at 16.6 MHz */
PRC = 8; /* Interrupt and Reset on M0 */
TOCD = 1; /* Timer0 Enable */

// VICEventAddr = (unsigned long)T0_IRQHandler; /* Set Interrupt Vector */
VICEventCtrl4 = 15; /* Lowest Priority for Timer0 Interrupt */
VICIntEnable = (1 << 4); /* Enable Timer0 Interrupt */

/* Enable and setup timer-1 interrupt, start timer */

// PCONP |= (1 << 2); /* Enable power to TIM block */
TICK1 = 280000; /* 1ms at 14.4 MHz */
PRC = 32; /* Interrupt and Reset on M0 */
TOCD = 3; /* Timer1 Enable */

// VICEventAddr = (unsigned long)T1_IRQHandler; /* Set Interrupt Vector */
VICEventCtrl1 = 0; /* Highest Priority than any other interrupt */
VICIntEnable = (1 << 5); /* Enable Timer1 Interrupt */

/* Enable and setup timer-2 interrupt, start timer */

// PCONP |= (1 << 2); /* Enable power to TIM block */
TICK3 = 480000; /* 1ms at 20 MHz */
PRC = 3; /* Timer2 Enable */

// VICIntSelect = (1 << 26); /* Timer2 as FIG Interrupt */
VICEventAddr2 = (unsigned long)T2_IRQHandler; /* Set Interrupt Vector */
VICEventCtrl1 = 3; /* More Priority than Timer0 Interrupt */
VICIntEnable = (1 << 26); /* Enable Timer2 Interrupt */

// Initialize LCD
lmdInit(tft_para);
lmdUnLock();
lmdId();
lmdUnLock();

// I2C_Init();
touch_init();

UARTInit(0); /* Init & Enable UART-0 */
UARTInit(1); /* Init & Enable UART-1 */

// ledTurnOn();
// LEDSetBacklight(100);

CaseStartup = 1;
TouchIntCtrl = 1;
```

Table 7.9 MAIN.C Part-3 listing

```c
RTC_Displ = 0; /* Stop displaying RTC in int. */
led_char_enable = 0; /* Stop displaying on led in int. */

load_picture(0, 0, 100, 400); /* Load the picture into RAM */
save_picture(0, 0, 100, 400); /* Save the picture into memory */
P_F甦福P_R=AL的p_AL=0=p_I_rms=p_Y_rms=p_Unit=1000; /* Display fresh value in int. */
P_FPMKDriver=P_FPW8Driver=P_Warning10 = P_Warning1 = 0; /* Set initial values */

// led_char_enable = 1; /* Start displaying on led in int. */
RTC_Displ = 1; /* Start displaying RTC in int. */

para_index = temp = 0;
while(1)
{
    if (temp == 1)
        SaveNorFlash(); /* Save receive data in Nor-Flash */

    if (temp == 2)
        break;

    if (para_index == 2)
    {
        if (ERRStatus == 0)
            /* Do something */

        while (WriteCmd0)
        {
            if (temp == 0)
                while (WriteCmd0)
                {
                    if (temp == 1)
                        while (WriteCmd0)
                        {
                            if (temp == 2)
                                while (WriteCmd0)
                                {
                                    if (temp == 3)
                                        while (WriteCmd0)
                                        {
```

7.3.3 Interrupt Routines (IRQ1.C)

The temperature of crucible is to be controlled through PID control loop, which is implemented as follows:

**Discrete-time PID Algorithm**

In Figure 7-4 a schematic of a system with a PID controller is shown. The PID controller compares the measured process value (crucible temperature) with a reference set point temperature. The difference or error is then processed to calculate a frequency burst. This frequency burst will try to adjust the measured process value back to the desired set point.

![Figure 7-4 Closed Loop System with PID controller](image)

Unlike simple control algorithms, the PID controller is capable of manipulating the process inputs based on the history and rate of change of the signal. This gives a more accurate and stable control method. The basic idea is that the controller reads the system state by a sensor. Then it subtracts the measurement from a desired reference to generate the error value. The error will be managed in three ways, to handle the present, through the proportional term, recover from the past, using the integral term, and to anticipate the future, through the derivate term.

Figure 7-5 shows the PID controller schematics, where $K_p$, $T_i$, and $T_d$ denote the constants of the proportional, integral, and derivative terms respectively.

![Figure 7.5 PID controller schematic](image)
Table 7.10 shows how this algorithm is coded in C.

```c
int Pid_Algorithm(int sv, int pv)
{
    int temp, retval, Error, p_term, i_term, d_term;

    // Proportionate
    Error = sv - pv;
    if(Error > Max_Error)
        p_term = MAX_INT;
    else if(Error < -Max_Error)
        p_term = -MAX_INT;
    else
        p_term = (Error * 256000)/P_Band;

    // Integral
    temp = PID_sum_error + Error;
    if(temp > Max_Sum_Error)
    {
        PID_sum_error = Max_Sum_Error;
        i_term = MAX_INT;
    }
    else if(temp < -Max_Sum_Error)
    {
        PID_sum_error = -Max_Sum_Error;
        i_term = -MAX_INT;
    }
    else
    {
        PID_sum_error = temp;
        i_term = (((PID_sum_error*2000)/I_Time)*128)/P_Band;
    }

    // Derivative
    d_term = (((PID_Last_pv-pv)*2000*D_Time)/P_Band)*128;
    if (d_term > MAX_INT)
        d_term = MAX_INT;
    else if (d_term < -MAX_INT)
        d_term = -MAX_INT;

    PID_Last_pv = pv;
    retval = (p_term+i_term+d_term)/128;

    if(retval > MAX_INT)
        retval = MAX_INT;
    else if(retval < -MAX_INT)
        retval = -MAX_INT;

    return(retval);
}
```

Table-7.10 C code for PID algorithm
## Autotuning Implementation

To transform the Ziegler-Nichols Frequency Domain (ZNFD) method to C code, the components needed are a relay, and a function to read the magnitude and frequency of oscillation. As the output of PID required to be generated is frequency bursts it can be considered as the relay operation. Following is the C code for functions to detect $a$ and $Tu$ and compute magnitude and frequency of oscillation, which are then utilized to compute $K_p$, $Ti$ & $Td$.

```c
//************* Global Variables ***********************
double p; // magnitude of P
double w; // frequency of P
double tu; // oscillation period ($\tau = 1/\omega$)
double d; // relay amplitude. This is constant for a particular relay
double s; // peak process output amplitude.
double yield; // keep previous process output
double ts; // sampling period
int i=0; // a counter to keep the number of iterations between two peaks

Thông tin code for functions to detect $a$ and $Tu$ and compute magnitude and frequency of oscillation, which are then utilized to compute $K_p$, $Ti$ & $Td$.

### Table-7.11 C code for computing a & Tu for Autotuning.

1. **void detect_a_tu(void)** // this must be a timer ISR running every ts seconds
2. { double y; // use to keep process output each sampling period
3.     y = read_input(); // read input from specified channel
4.     if (y>0) a = y; // compare new input with a, if greater keep it as new a
5.     if (yold<0 && y>0)
6.         { // detect zero crossing
7.             tu = tu+ts;
8.             i=0;
9.         }
10.     yold = y;
11.     i++;
12. }
13. void compute_pw(void) // run this after we get values for a and tu
14. { double Na;
15.     Na = (4*a)/pi*s;
16.     p = -1/Na; // gain of $P(j\omega)$ at point of intersection
17.     w = 2*pi/tu; // frequency in rad/s
18. }
```

7.3.4 Serial Port Routines (UART.C)

The MOD-BUS communication & reading & writing of machine parameters from & to micro-controller board is carried out in this section as listed in Table 7.12.

```c
if (ReadAddr == 0x0000)
{
    Sp = (int)((UART0Buffer[3]<<5) + UART0Buffer[4]);
    ALSp = (int)((UART0Buffer[1]<<5) + UART0Buffer[6]);
    ALMy = (int)UART0Buffer[8];
}
else if (ReadAddr == 0x0003)
{
    Var[PIDFAKE][1] = (int)((UART0Buffer[3]<<8) + UART0Buffer[4]);
    Var[PIDFAKE][2] = (int)((UART0Buffer[5]<<8) + UART0Buffer[6]);
    Var[PIDFAKE][3] = (int)((UART0Buffer[7]<<8) + UART0Buffer[8]);
    ReadPBTTID = 9;
}
else
{
    if (UART0Buffer[4] != 3)
        AutoTuneON = 0;
}
else if (UART0Buffer[1] == 1)
{
    switch(ReadAddr)
    {
        case 0x0003:
            OFStatus = UART0Buffer[3];
            ERStatus = UART0Buffer[4];
            break;
        case 0x0004:
            ALStatus = UART0Buffer[4];
            break;
        default:
            if (UART0Buffer[3]&0x80)
                PV = 0;
            else
                PV = (int)((UART0Buffer[3]<<8) + UART0Buffer[4]);
            if (UART0Buffer[5]&0x80)
                I_zms = 0;
            else
                I_zms = (int)((UART0Buffer[5]<<8) + UART0Buffer[6]);
            if (UART0Buffer[7]&0x80)
                V_zms = 0;
            else
                V_zms = (int)((UART0Buffer[7]<<8) + UART0Buffer[8]);
    }
}
```

Table-7.12 UART.C subroutines listing

7.4 Summary

In this chapter the software implementation for micro-controller board as well as ARM-7 board is described. The micro-controller board software automatically tunes the gate firing of IGBT according to the changes in load inductance. ARM-7 board software takes care of graphical user interface to view/insert machine parameters. The PID & Autotuning is explained with respect to ARM-7 controller.