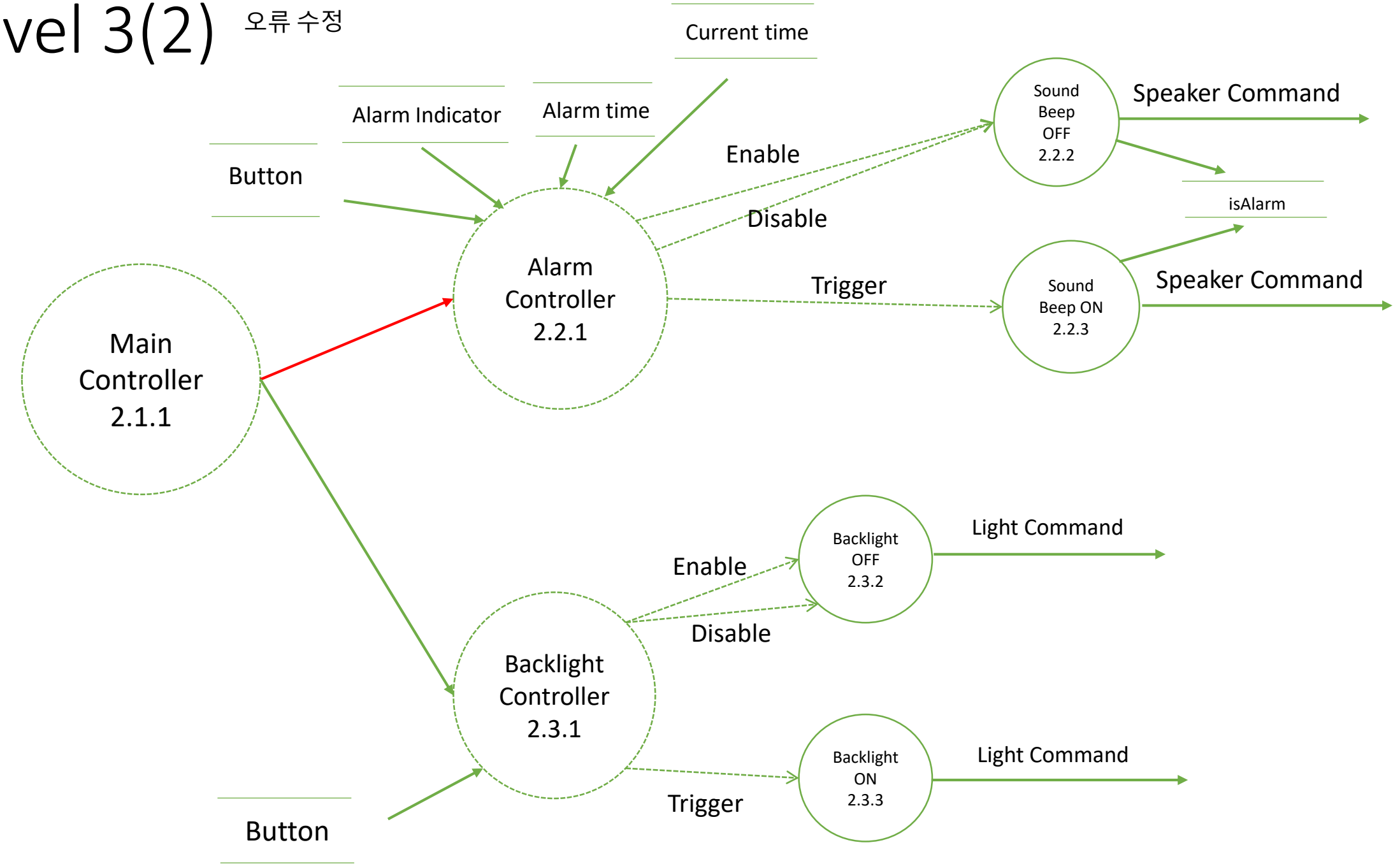
Software Engineering

Digital Watch System – SA

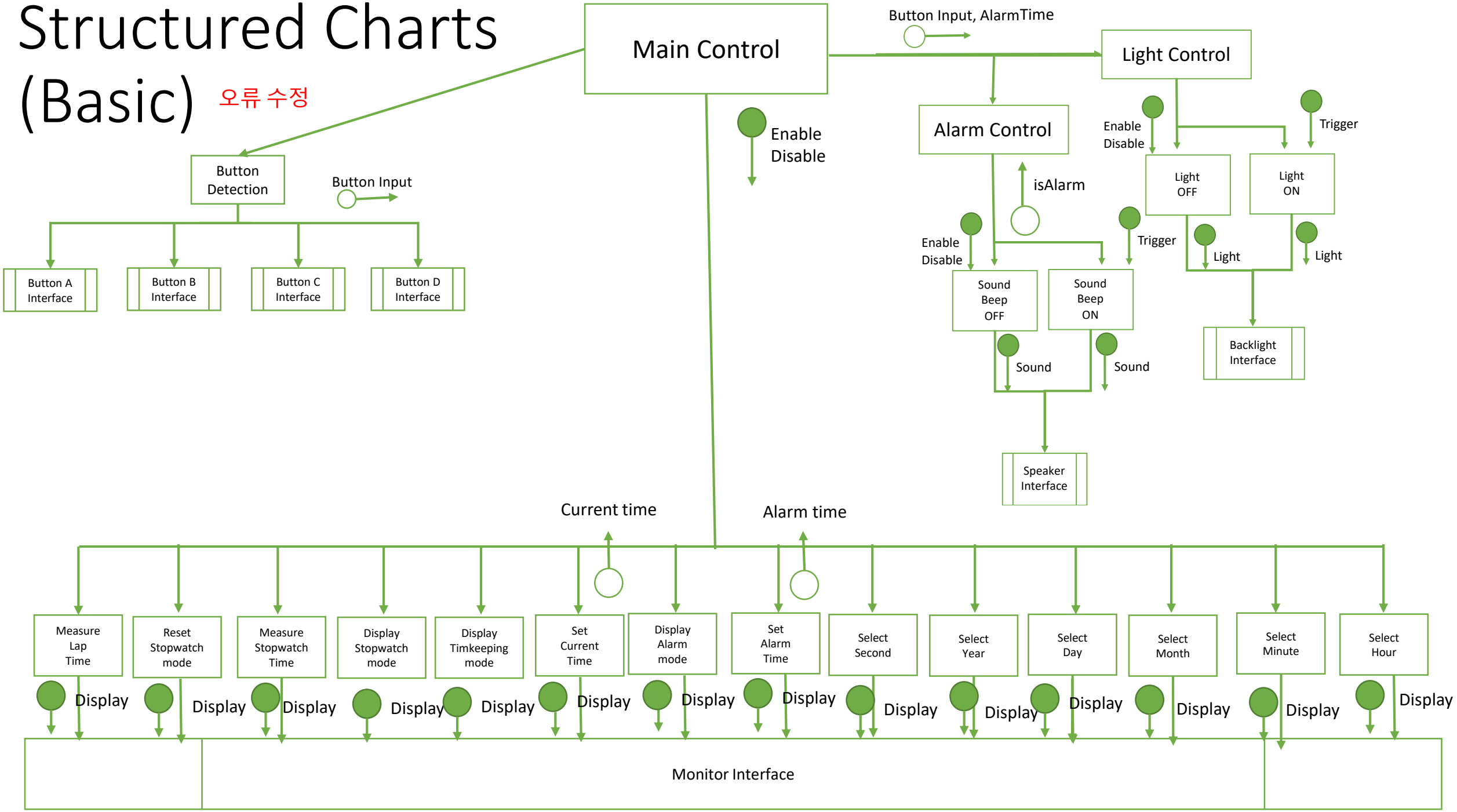
TEAM2.
201611251 공민정
201611261 민지호
201611276 이규은
201611309 최지현

DFD Level 3(2) 오류 수정



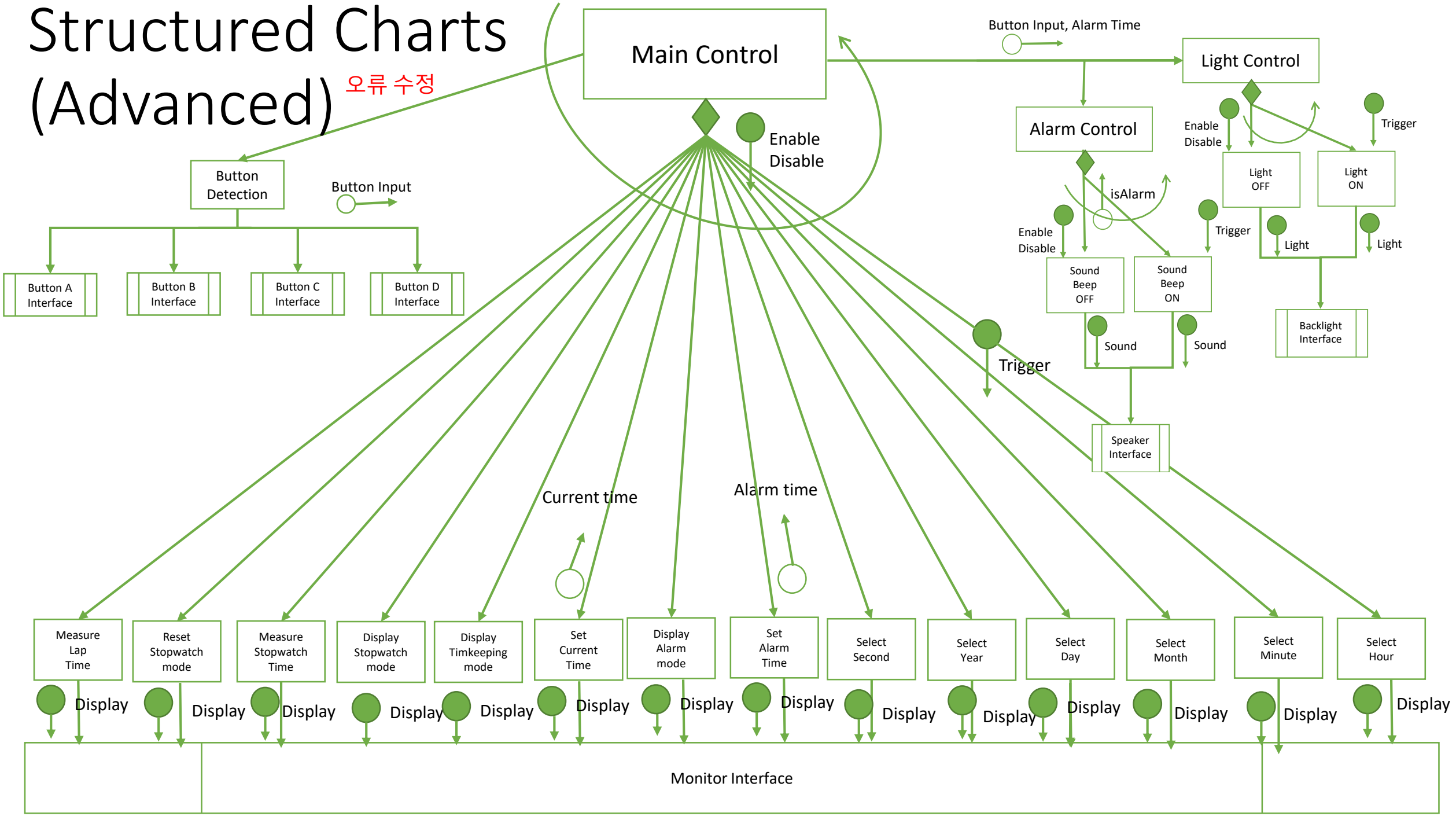
Structured Charts (Basic)

오류 수정



Structured Charts (Advanced)

오류 수정

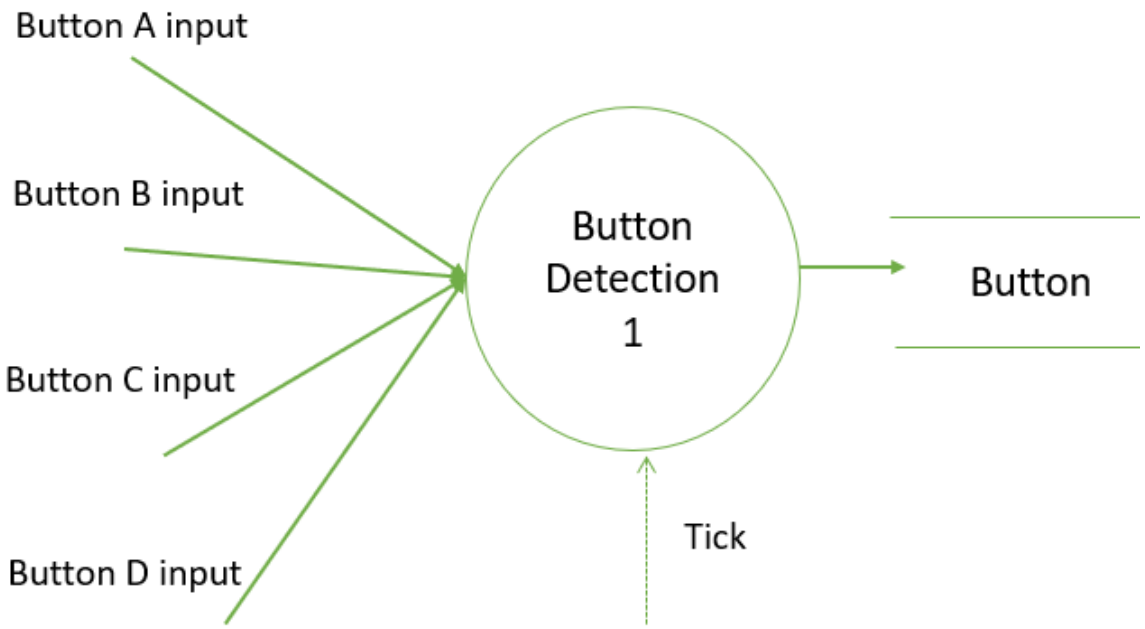


Header

```
#define BUFSIZE 4


/* define Modes */
#define TKMODE 10
#define ALMODE 20
#define SWMODE 30

#define STCURTIME 11

#define STSEC 12
#define STMIN 13
#define STHOU 14
#define STDAY 15
#define STMON 16
#define STYEA 17


#define STALRTIME 21

#define MSSWTIME 31 // measure stopwatch time
#define MSLPTIME 32 // measure laptime
#define RSSWTIME 33 // reset stopwatch time


#define BUTTONA 'A'
#define BUTTONB 'B'
#define BUTTONC 'C'
#define BUTTOND 'D'
```

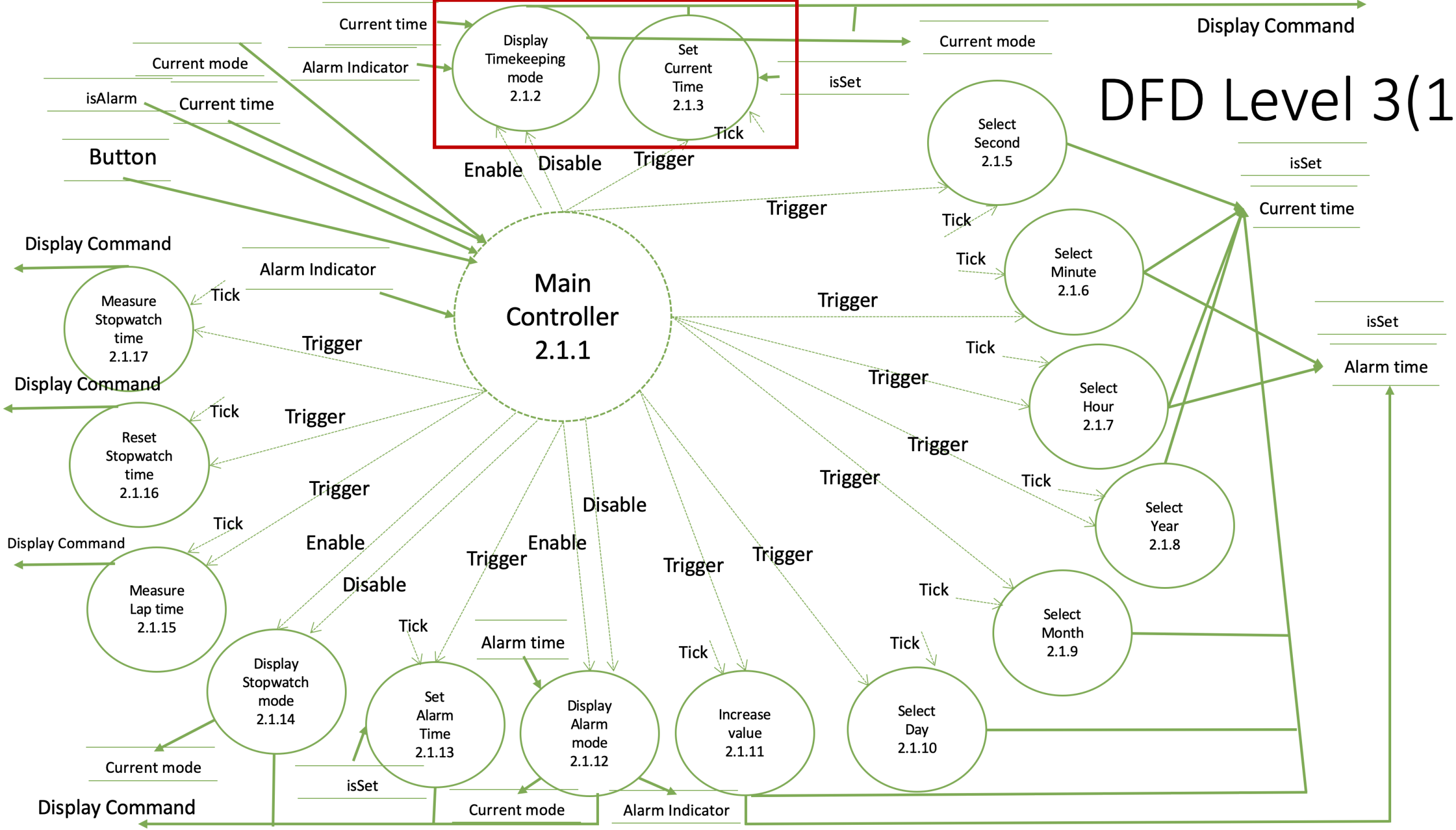
```
typedef enum boolean { false, true } bool;
typedef struct tm s_tm;

extern int month_day[12];
void moon_year(s_tm *t);

bool alarmCheck(bool *isAlarm, char *input, struct tm *currentTime, struct tm *alarmTime);
void backlightCheck(bool isOff);
int kbhit(void);
void setTime(s_tm *t);
void determinePriority(char* inputBuffer);
```

Determine Priority




```
void determinePriority(char *input){
    char temp = input[0]+16;
    for(int i=1; i<BUFSIZE; i++){
        input[i]+=16;
        if(input[i] > temp){
            temp = input[i];
        }
    }
    input[0] = temp;
    return;
}
```

DFD Level 3(1)


Main Controller 2.1.1
Display Timekeeping mode 2.1.2
Set Current Time 2.1.3
Select Second 2.1.5
Select Minute 2.1.6
Select Hour 2.1.7
Select Year 2.1.8
Select Month 2.1.9
Select Day 2.1.10
Increase value 2.1.11
Display Alarm mode 2.1.12
Set Alarm Time 2.1.13
Display Stopwatch mode 2.1.14
Measure Lap time 2.1.15
Reset Stopwatch time 2.1.16
Measure Stopwatch time 2.1.17
Current time
Alarm Indicator
Current mode
isAlarm
Button
Display Command
isSet
Alarm time
Enable
Disable
Trigger
Tick

Main Controller – Time keeping

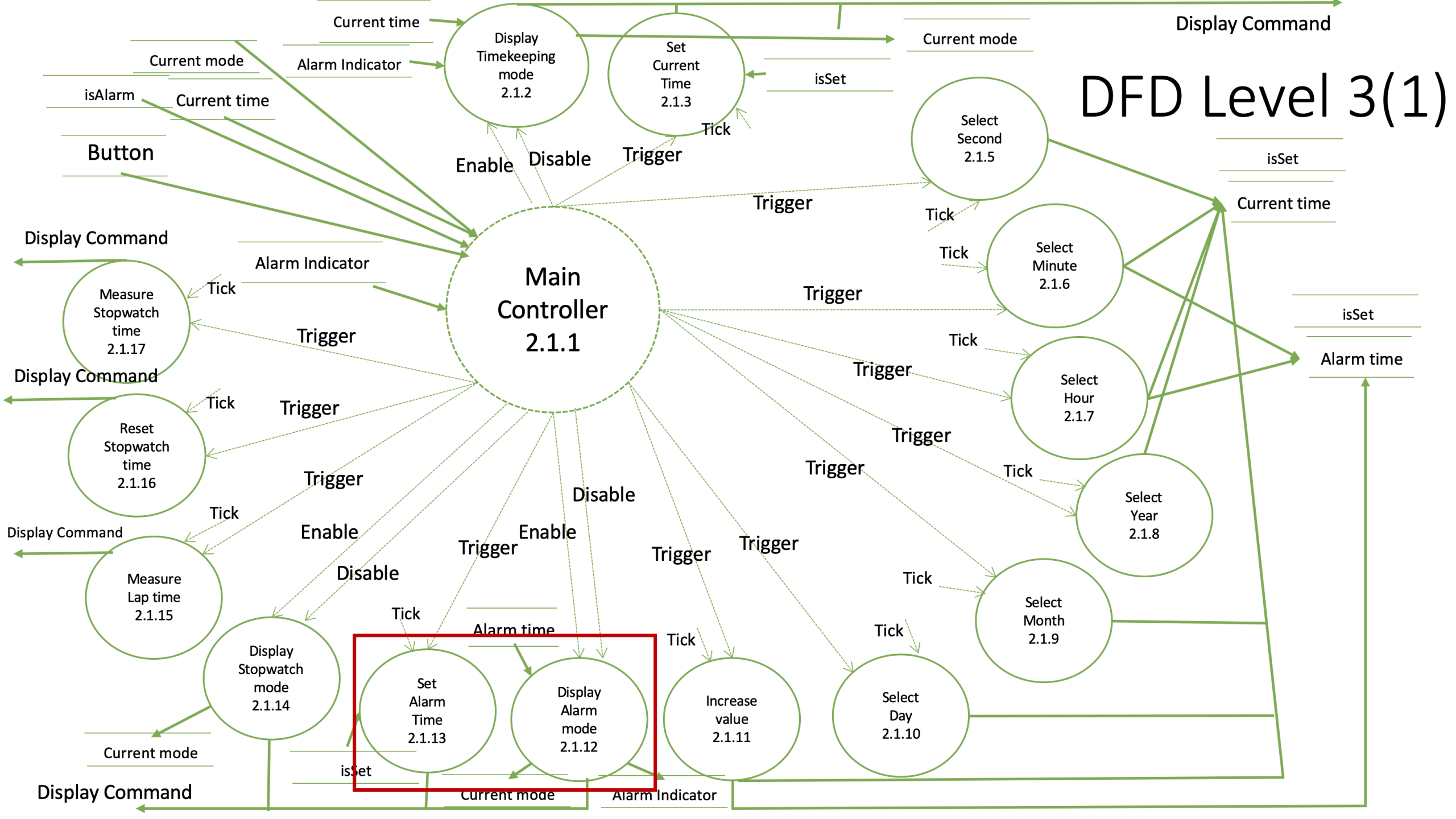
```
switch (status) {
case TKMODE:
        if (input[0] == BUTTONC) {
                status = ALMODE;

        }
        else if (input[0] == BUTTONA) {
                status = STCURTIME;
        }

        display();
        break;

case STCURTIME: /* set current time */
        if (isSet == status) {
                isSet = '\0';
                status = TKMODE;
        }
        else {
                isSet = status; /* mark as timekeeping mode */
                status = STSEC;
        }
        display();
        break;
```

Main Controller – Set Time

```
case STMIN:
        if (input[0] == BUTTONA) { /* return to display mode */
                status = isSet;

        }
        else if (input[0] == BUTTONB) { /* increase minutes */
                if (isSet == STCURTIME) {
                        currentTime->tm_min++;
                        setTime(currentTime);
                }

                else if (isSet == STALRTIME) {
                        alarmTime->tm_min++;
                        setTime(alarmTime);
                }
        }
        else if (input[0] == BUTTONC) { /* change digit hour > minute or hour > day */
                if (isSet == STCURTIME) {
                        status = STDAY;
                }
                else if (isSet == STALRTIME) {
                        status = STHOU;
                }
        }
        display();
        break;
```

```
case STYEA:
        if (input[0] == BUTTONA) { /* return to display mode */
                status = isSet;
        }
        else if (input[0] == BUTTONB) { /* increase year */
                int sum_day = 0;
                moon_year(currentTime);

                for (int i = currentTime->tm_mon+1; i <= 12; i++)
                        sum_day += month_day[i-1];

                currentTime->tm_year++;
                moon_year(currentTime);

                for(int i=1;i<currentTime->tm_mon+1;i++)
                        sum_day += month_day[i-1];

                currentTime->tm_wday += sum_day;
                setTime(currentTime);

        }
        else if (input[0] == BUTTONC) { /* change digit year > second */
                status = STSEC;
        }
        display();
        break;
```

DFD Level 3(1)

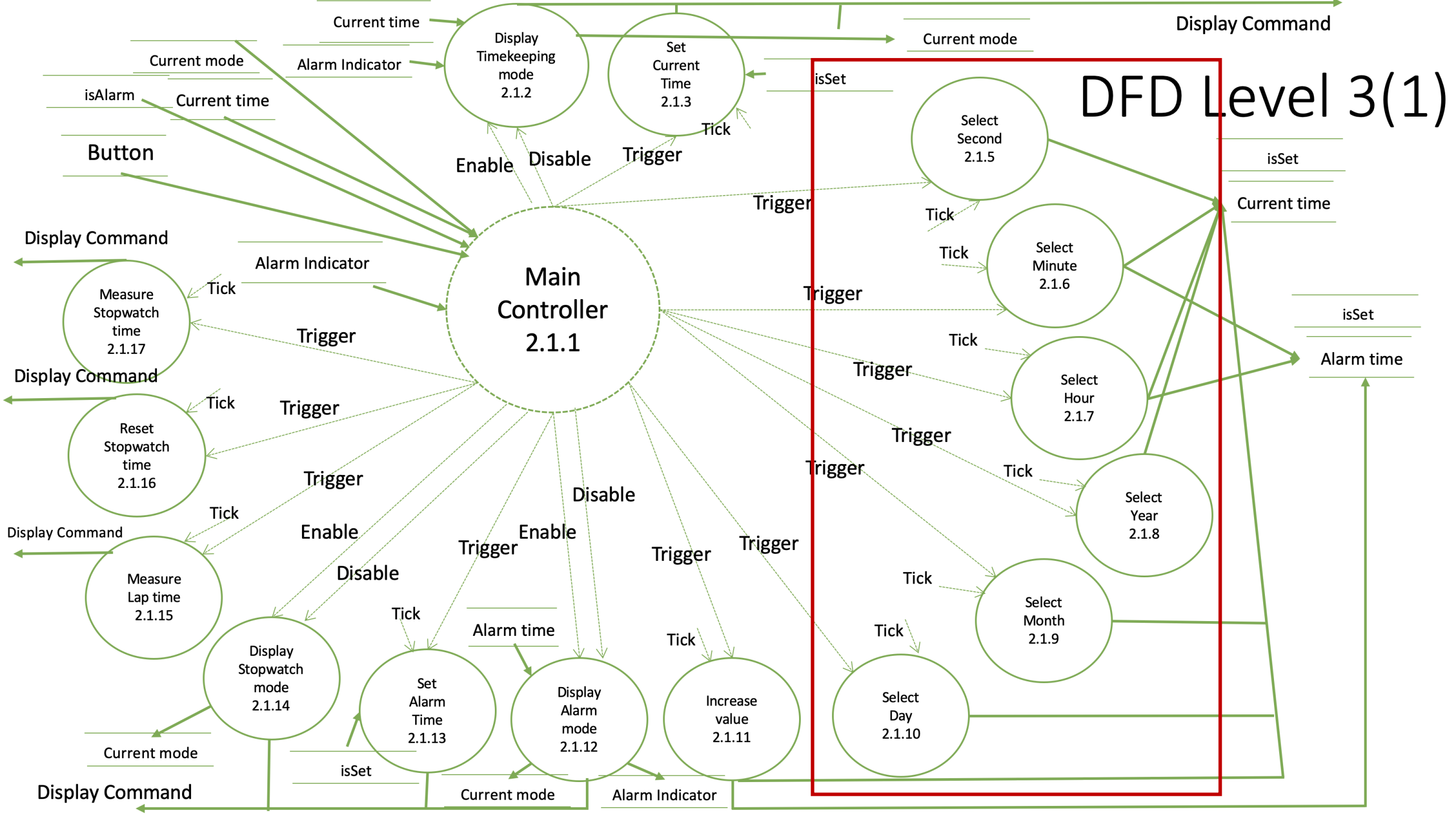

Current time
Display Timekeeping mode 2.1.2
Set Current Time 2.1.3
Display Command
Current mode
isSet
Current mode
Alarm Indicator
isAlarm
Current time
Button
Tick
Enable
Disable
Trigger
Select Second 2.1.5
isSet
Current time
Trigger
Tick
Tick
Select Minute 2.1.6
Display Command
Alarm Indicator
Main Controller 2.1.1
Trigger
isSet
Alarm time
Measure Stopwatch time 2.1.17
Tick
Trigger
Tick
Trigger
Select Hour 2.1.7
Display Command
Reset Stopwatch time 2.1.16
Tick
Trigger
Trigger
Trigger
Tick
Select Year 2.1.8
Trigger
Disable
Tick
Display Command
Enable
Enable
Trigger
Trigger
Trigger
Tick
Measure Lap time 2.1.15
Disable
Select Month 2.1.9
Tick
Tick
Alarm time
Tick
Tick
Display Stopwatch mode 2.1.14
Set Alarm Time 2.1.13
Display Alarm mode 2.1.12
Increase value 2.1.11
Select Day 2.1.10
Current mode
isSet
Current mode
Alarm Indicator
Display Command

Main Controller – Alarm

```
case ALMODE:
        if (input[0] == BUTTONC) {
                status = SWMODE;
        }
        else if (input[0] == BUTTONB) {
                alarmIndicator = !alarmIndicator;
        }
        else if (input[0] == BUTTONA) {
                status = STALRTIME;
        }
        display();
        break;

case STALRTIME: /* set alarm time */
        if (isSet == status) {
                isSet = '\0';
                status = ALMODE;
        }
        else {
                isSet = status; /* mark as alarm mode */
                status = STHOU;
        }
        display();
        break;
```

Main Controller – Set Time

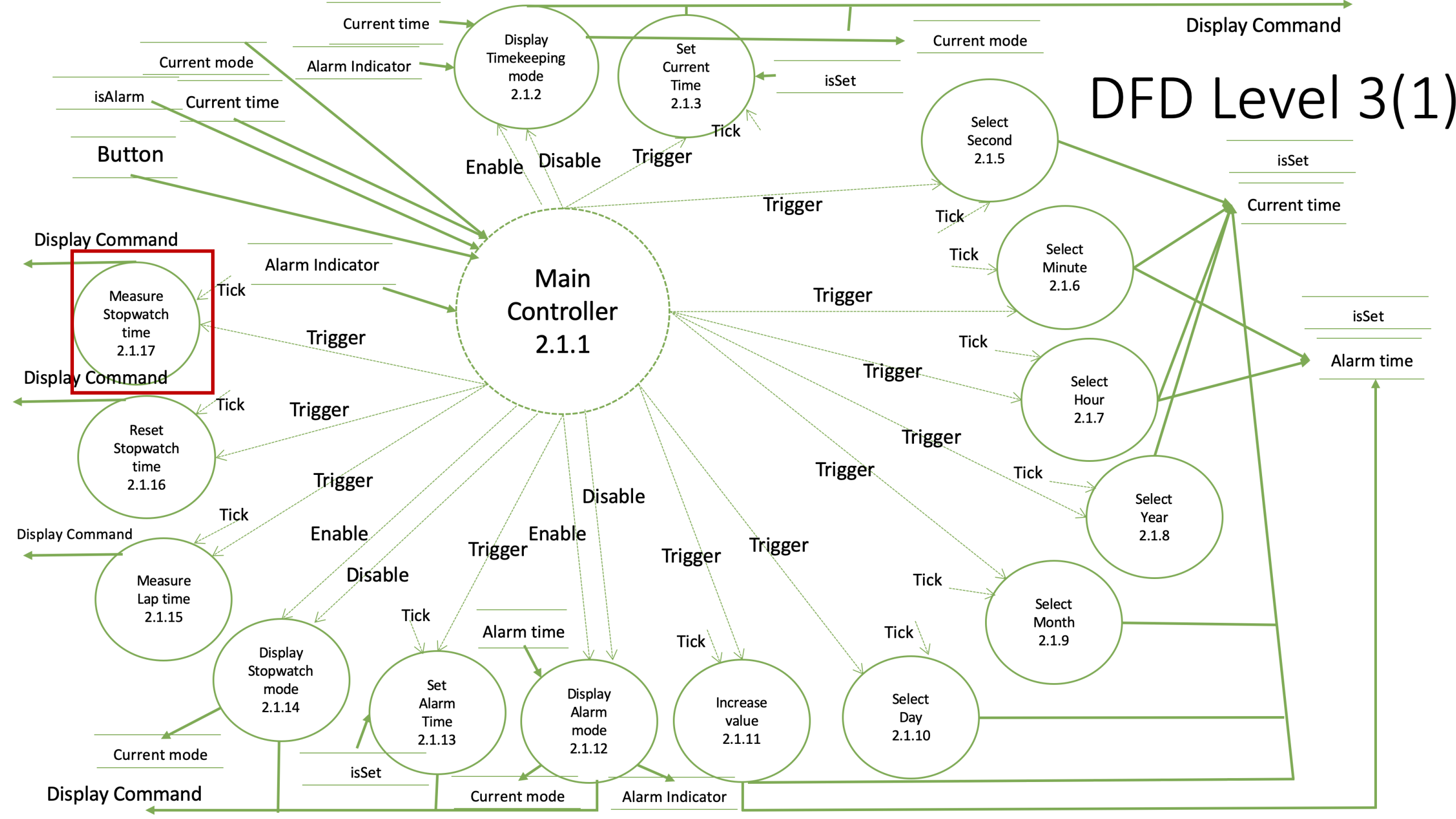
```
case STMIN:
        if (input[0] == BUTTONA) { /* return to display mode */
                status = isSet;

        }
        else if (input[0] == BUTTONB) { /* increase minutes */
                if (isSet == STCURTIME) {
                        currentTime->tm_min++;
                        setTime(currentTime);
                }

                else if (isSet == STALRTIME) {
                        alarmTime->tm_min++;
                        setTime(alarmTime);
                }
        }
        else if (input[0] == BUTTONC) { /* change digit hour > minute or hour > day */
                if (isSet == STCURTIME) {
                        status = STDAY;
                }
                else if (isSet == STALRTIME) {
                        status = STHOU;
                }
        }
        display();
        break;
```


DFD Level 3(1)
Main Controller 2.1.1
Display Timekeeping mode 2.1.2
Set Current Time 2.1.3
Measure Stopwatch time 2.1.17
Reset Stopwatch time 2.1.16
Measure Lap time 2.1.15
Display Stopwatch mode 2.1.14
Set Alarm Time 2.1.13
Display Alarm mode 2.1.12
Increase value 2.1.11
Select Second 2.1.5
Select Minute 2.1.6
Select Hour 2.1.7
Select Year 2.1.8
Select Month 2.1.9
Select Day 2.1.10
Current time
Alarm Indicator
Current mode
isAlarm
Button
Display Command
isSet
Alarm time
Tick
Trigger
Enable
Disable

Main Controller - Select ” “

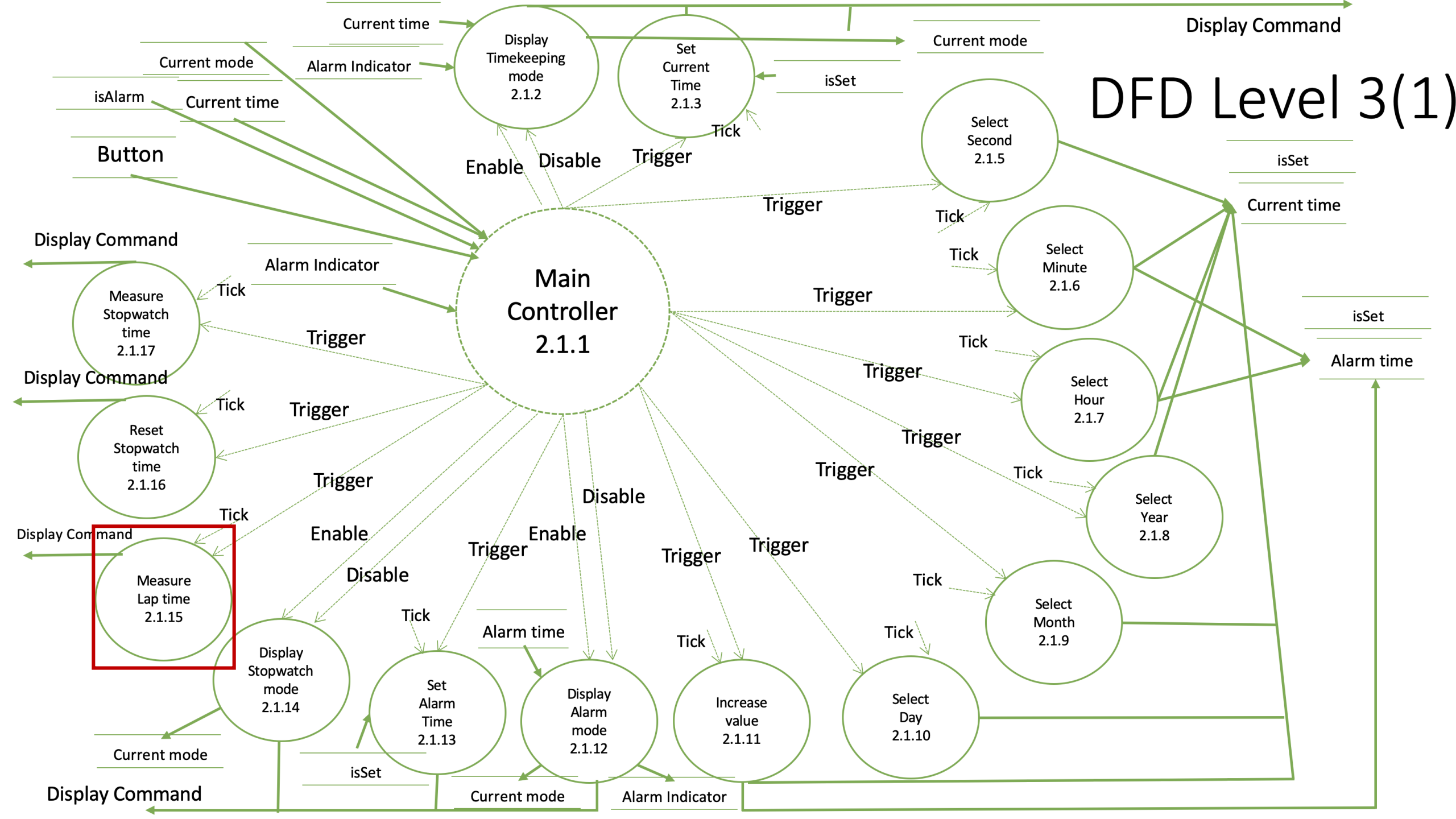
```
if (t->tm_sec >= 60) {
    t->tm_sec = 0;
    t->tm_min++;
}
if (t->tm_min >= 60) {
    t->tm_min = 0;
    t->tm_hour++;
}
if (t->tm_hour >= 24) {
    t->tm_hour = 0;
    t->tm_mday++;
    t->tm_wday++;

}
moon_year(t);
if (t->tm_mday > month_day[t->tm_mon]) {
    t->tm_wday += month_day;
    t->tm_mday = 1;
    t->tm_mon++;
}
if (t->tm_mon > 12) {
    t->tm_mon = 1;
    t->tm_year++;
}
if (t->tm_year >= 2100) {
    t->tm_year = 2019;
}

t->tm_wday %= 7;
```

```
void moon_year(s_tm *t) {
    if ((t->tm_year % 4 == 0 && t->tm_year % 100) || t->tm_year % 400 == 0) {
        month_day[1] = 29;
    }
    else
        month_day[1] = 28;

}
```


DFD Level 3(1)
Main Controller 2.1.1
Display Timekeeping mode 2.1.2
Set Current Time 2.1.3
Select Second 2.1.5
Select Minute 2.1.6
Select Hour 2.1.7
Select Year 2.1.8
Select Month 2.1.9
Select Day 2.1.10
Increase value 2.1.11
Display Alarm mode 2.1.12
Set Alarm Time 2.1.13
Display Stopwatch mode 2.1.14
Measure Lap time 2.1.15
Reset Stopwatch time 2.1.16
Measure Stopwatch time 2.1.17
Current time
Current mode
Alarm Indicator
isAlarm
Button
Display Command
isSet
Alarm time
Enable
Disable
Trigger
Tick

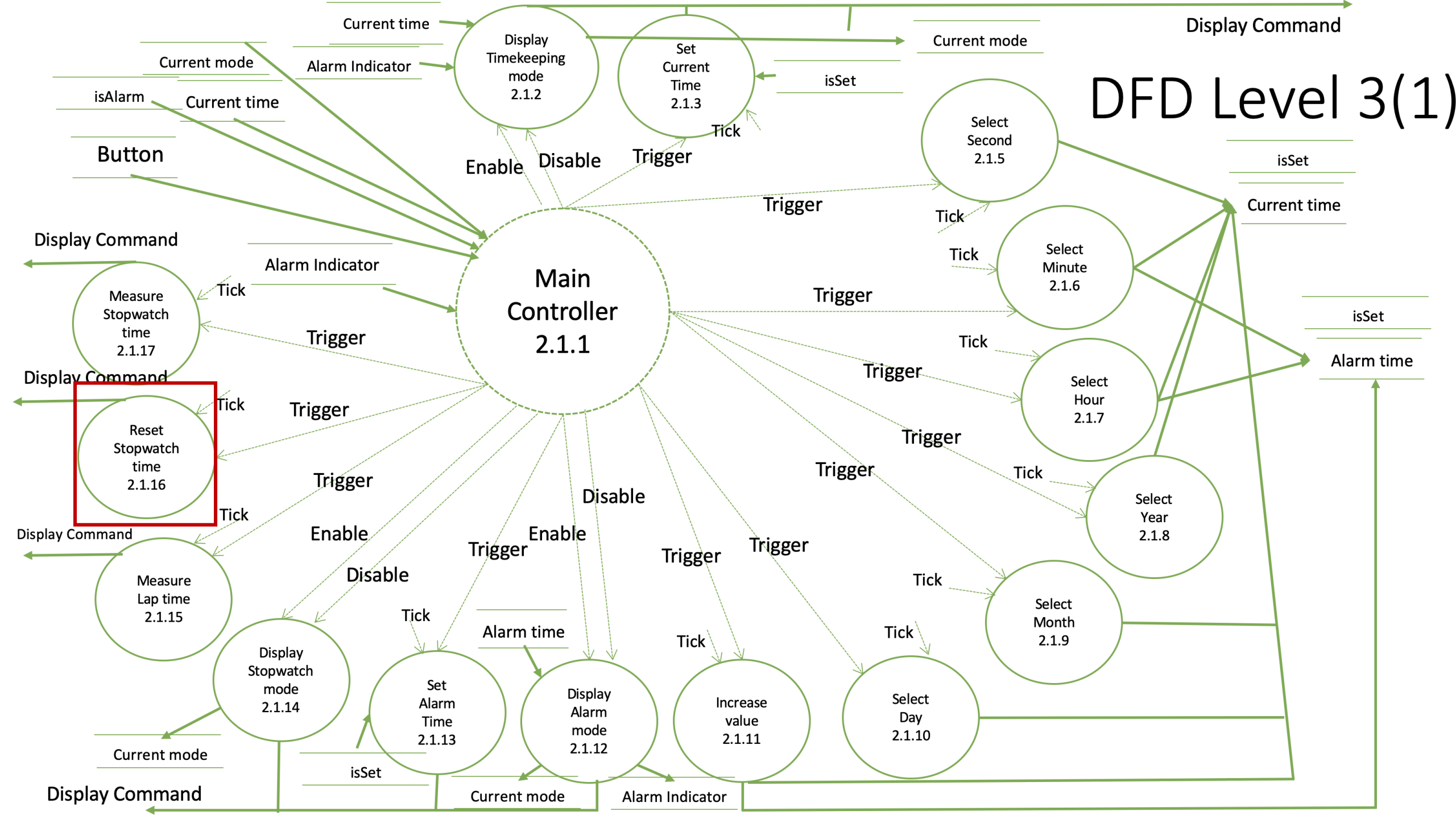
Main Controller – Measure Stopwatch Time

```
case MSSWTIME: /* measure stopwatch time */
        sw_end = clock();
        measureTime = (double)( (sw_end - sw_start) *100) / CLOCKS_PER_SEC;
        while (measureTime > 99){
                swTime->tm_sec++;
                measureTime-=99;
        }
        sw_start = clock();
        setTime(swTime);

        if (input[0] == BUTTONA) { /* measure laptime */
                status = MSLPTIME;
                lapTime = measureTime;

                lpTime->tm_sec = swTime->tm_sec;
                lpTime->tm_min = swTime ->tm_min;
        }
        else if (input[0] == BUTTONB) { /* add up measure time */
                status = SWMODE;
        }

        display();
        break;
```


DFD Level 3(1)
Main Controller 2.1.1
Display Timekeeping mode 2.1.2
Set Current Time 2.1.3
Select Second 2.1.5
Select Minute 2.1.6
Select Hour 2.1.7
Select Year 2.1.8
Select Month 2.1.9
Select Day 2.1.10
Increase value 2.1.11
Display Alarm mode 2.1.12
Set Alarm Time 2.1.13
Display Stopwatch mode 2.1.14
Measure Lap time 2.1.15
Reset Stopwatch time 2.1.16
Measure Stopwatch time 2.1.17
Current time
Alarm Indicator
Current mode
isAlarm
Button
Display Command
isSet
Alarm time
Enable
Disable
Trigger
Tick

Main Controller – Measure Lap Time

```
case MSLPTIME: /* measure laptime */
        sw_end = clock();
        measureTime = (double)( (sw_end - sw_start) *100) / CLOCKS_PER_SEC;
        while (measureTime > 99){
                swTime->tm_sec++;
                measureTime-=99;
        }
        sw_start = clock();
        setTime(swTime);

        if (input[0] == BUTTONA) {
                lapTime = measureTime;

                lpTime->tm_sec = swTime->tm_sec;
                lpTime->tm_min = swTime ->tm_min;

                display();

        }
        else if (input[0] == BUTTONB) {
                status = MSSWTIME;

                display();
        }
        break;
```

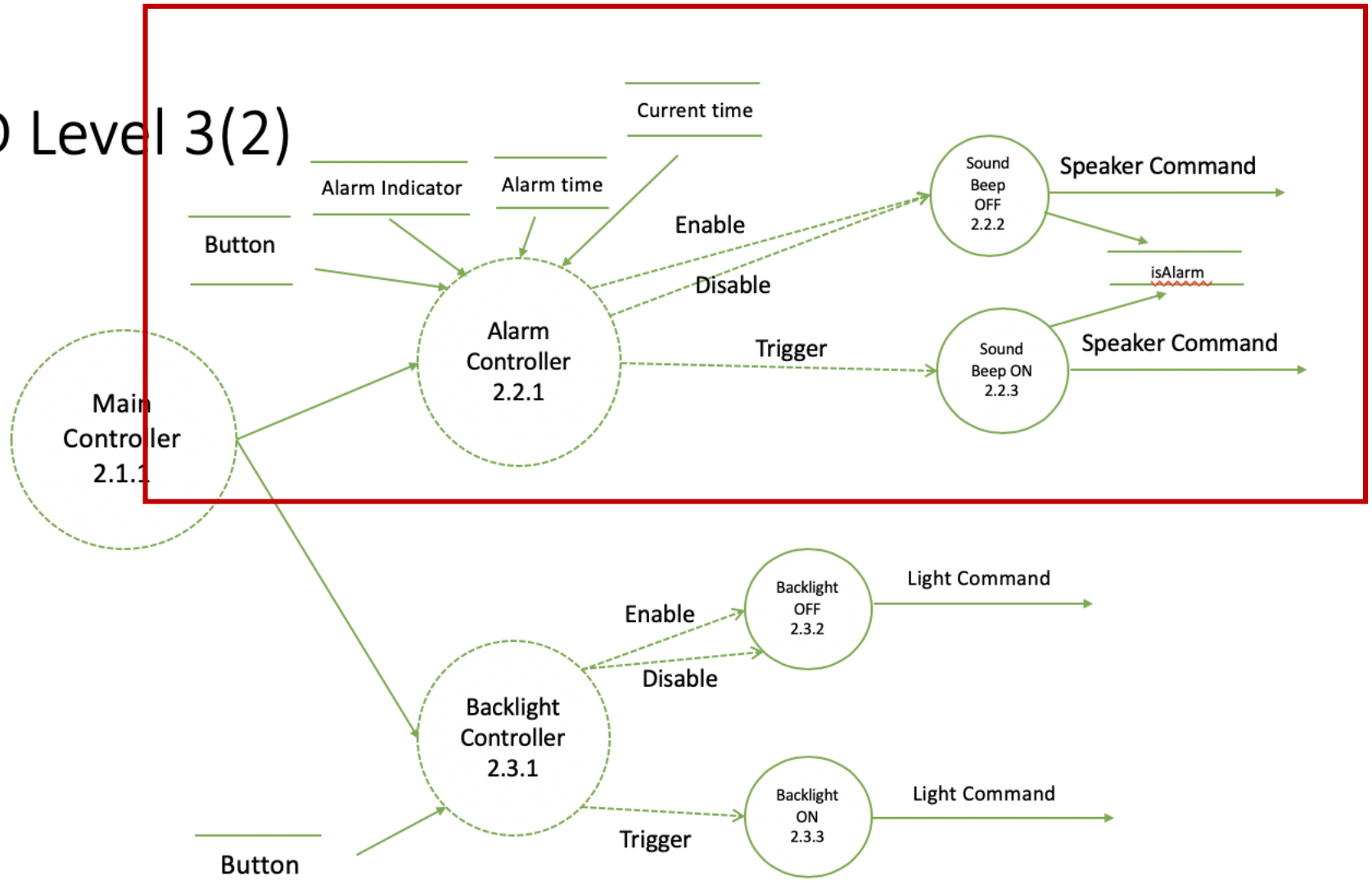

DFD Level 3(1)
Main Controller 2.1.1
Display Timekeeping mode 2.1.2
Set Current Time 2.1.3
Select Second 2.1.5
Select Minute 2.1.6
Select Hour 2.1.7
Select Year 2.1.8
Select Month 2.1.9
Select Day 2.1.10
Increase value 2.1.11
Display Alarm mode 2.1.12
Set Alarm Time 2.1.13
Display Stopwatch mode 2.1.14
Measure Lap time 2.1.15
Reset Stopwatch time 2.1.16
Measure Stopwatch time 2.1.17
Button
isAlarm
Current time
Current mode
Alarm Indicator
Display Command
isSet
Alarm time
Enable
Disable
Trigger
Tick

Main Controller – Reset Stopwatch Time

```
case SWMODE:
        if (input[0] == BUTTONA) {
                status = RSSWTIME;
                measureTime = 0;
                lapTime = 0;
```

```
case RSSWTIME: /* reset stopwatch time */
        status = SWMODE;

        display();
        break;
```

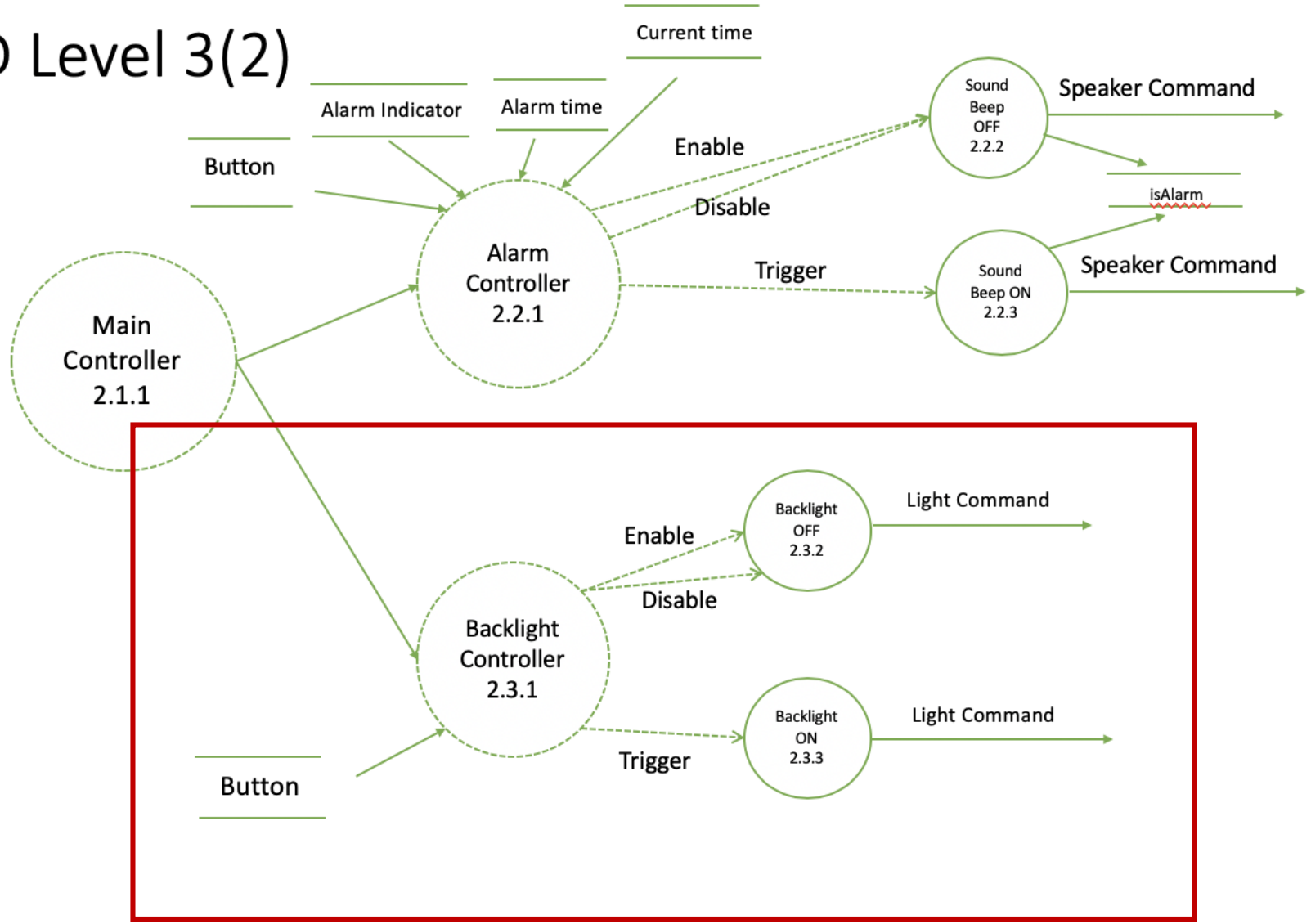
DFD Level 3(2)


Current time
Alarm Indicator
Alarm time
Button
Enable
Disable
Sound Beep OFF 2.2.2
Speaker Command
isAlarm
Alarm Controller 2.2.1
Trigger
Sound Beep ON 2.2.3
Speaker Command
Main Controller 2.1.1
Backlight OFF 2.3.2
Light Command
Enable
Disable
Backlight Controller 2.3.1
Backlight ON 2.3.3
Light Command
Trigger
Button

Alarm Controller

```
bool alarmCheck(bool *isAlarm, char* input, struct tm *currentTime, struct tm *alarmTime) {
        if (*isAlarm) {
                fprintf(stdout, "\aBeep!\n");
                if (input[0] == BUTTONA || input[0] == BUTTONB || input[0] == BUTTONC || input[0] == BUTTOND) {
                        input[0] = 0;
                        *isAlarm = false; /* alarm OFF */
                        return false;
                }
        }
        else {
                if ((currentTime->tm_hour == alarmTime->tm_hour) && (currentTime->tm_min == alarmTime->tm_min)) {
                        *isAlarm = true;
                        return true;
                }
        }
        return true;
}
```

DFD Level 3(2)


Current time
Alarm Indicator
Alarm time
Button
Enable
Disable
Sound Beep OFF 2.2.2
Speaker Command
isAlarm
Alarm Controller 2.2.1
Trigger
Sound Beep ON 2.2.3
Speaker Command
Main Controller 2.1.1
Backlight OFF 2.3.2
Light Command
Enable
Disable
Backlight Controller 2.3.1
Backlight ON 2.3.3
Light Command
Trigger
Button

Backlight Controller

```
else { /* if alarm is off, execute enabled/triggered process */

        if (input[0] == BUTTOND) {
                light_start = clock();
                isOff = false;
        }
        else {
                if (!isOff) {
                        light_end = clock();

                        if ((double)(light_end - light_start) / CLOCKS_PER_SEC > 4) {
                                isOff = true;
                        }
                }
                backlightCheck(isOff);
```

```
void backlightCheck(bool isOff){
    if(isOff){
        printf("\033[0m");
    } else {
        printf("\033[1;33m");
    }
    return;
}
```

Thank you!