

COE 538 - Robot Guidance Challenge
Final Report

Lab Section	02
-------------	----

Student Name	Student Number
Jonathan Ma	500837227
Stanley Tan	501017455

Code Explanation

Entry:	ORG	\$4000
_Startup:	LDS	#\$4000
	JSR	initPORTS
	JSR	initAD
	JSR	initLCD
	JSR	clrLCD
	JSR	initTCNT
	CLI	
	LDX	#msg1
	JSR	putsLCD
	LDAA	#\$8A
	JSR	cmd2LCD
	LDX	#msg2
	JSR	putsLCD
	LDAA	#\$C0
	JSR	cmd2LCD
	LDX	#msg3
	JSR	putsLCD
	LDAA	#\$C7
	JSR	cmd2LCD
	LDX	#msg4
	JSR	putsLCD
main	JSR	UPDT_READING
	JSR	UPDT_DISPL
	LDAA	CRNT_STATE
	JSR	DISPATCHER
	BRA	main

This is the startup code and the main code. The startup code initializes several functions which the eebot requires. The main code runs the same 4 lines repeatedly to execute subsections which carry out the code.

DISPATCHER	CMPA	#START		FWD_ST	PULD	
	BNE	NOT_START			BRSET	PORTAD0,\$04,NO_FWD_BUMP
	JSR	START_ST			LDAA	SEC_PTH_INT
	RTS				STAA	NEXT_D
NOT_START	CMPA	#FWD			JSR	INIT_REV
	BNE	NOT_FORWARD			MOVB	#REV,CRNT_STATE
	JMP	FWD_ST			JMP	FWD_EXIT
NOT_FORWARD	CMPA	#RT_TRN		NO_FWD_BUMP	BRSET	PORTAD0,\$08,NO_REV_BUMP
	BNE	NOT_RT_TRN			JMP	INIT_BK_TRK
	JSR	RT_TRN_ST			MOVB	#BK_TRK,CRNT_STATE
	RTS				JMP	FWD_EXIT
NOT_RT_TRN	CMPA	#LT_TRN		NO_REV_BUMP	LDAA	SENS_C
	BNE	NOT_LT_TRN			BEQ	NO_RT_INTXN
	JSR	LT_TRN_ST			LDAA	NEXT_D
	RTS				PSHA	
NOT_LT_TRN	CMPA	#REV			LDAA	PRI_PTH_INT
	BNE	NOT_REVERSE			STAA	NEXT_D
	JSR	REV_ST			JSR	INIT_RT_TRN
	RTS				MOVB	#RT_TRN,CRNT_STATE
NOT_REVERSE	CMPA	#BK_TRK		NO_RT_INTXN	JMP	FWD_EXIT
	BNE	NOT_BK_TRK			LDAA	SENS_B
	JMP	BK_TRK_ST			BEQ	NO_LT_INTXN
NOT_BK_TRK	CMPA	#SBY			LDAA	SENS_A
	BNE	NOT_SBY			BEQ	LT_TURN
	JSR	SBY_ST			LDAA	NEXT_D
	RTS				PSHA	
NOT_SBY	NOP				LDAA	PRI_PTH_INT
DISP_EXIT	RTS				STAA	NEXT_D
					BRA	NO_SHFT_LT
					LDAA	NEXT_D
					PSHA	
					LDAA	SEC_PTH_INT
					STAA	NEXT_D
					JSR	INIT_LT_TRN
					MOVB	#LT_TRN,CRNT_STATE
					JMP	FWD_EXIT
				NO_LT_INTXN	LDAA	SENS_C
					BEQ	NO_SHFT_RT
					JSR	PORTON
					LDD	COUNT2
					CPD	#INC_DIS
					BLO	RT_FWD_DIS
					JSR	INIT_FWD
					JMP	FWD_EXIT
				RT_FWD_DIS	LDAA	SENS_C
					BEQ	NO_SHFT_LT
					JSR	STARON
					LDD	COUNT1
					CPD	#INC_DIS
					BLO	LT_FWD_DIS
					JSR	INIT_FWD
					JMP	FWD_EXIT
				NO_SHFT_RT	LDAA	SENS_C
					BEQ	NO_SHFT_LT
					JSR	STARON
					LDD	COUNT1
					CPD	#FWD_DIS
					BLO	FWD_STR_DIS
					JSR	INIT_FWD
					JMP	FWD_EXIT
				NO_SHFT_LT	JSR	STARON
					JSR	PORTON
					LDD	COUNT1
					CPD	#FWD_DIS
					BLO	FWD_STR_DIS
					JSR	INIT_FWD
					JMP	FWD_EXIT
				FWD_EXIT	JMP	main

This is the Dispatcher code. This helps the eebot identify which process to run next. The code beside the Dispatcher tells the eebot to follow the black line which alternates between different states of rotation.

START_ST	BRCLR	PORTAD0,\$04,NO_FWD
	JSR	INIT_FWD
	MOVB	#FWD,CRNT_STATE
	BRA	START_EXIT
NO_FWD	NOP	
START_EXIT	RTS	

This is the start state. When the front bumper is pressed, the bot will begin to start and run.

COE 538 Robot Guidance Challenge

REV_ST	LDD	COUNT1	RT_TRN_ST	LDD	COUNT2
	CPD	#REV_DIS		CPD	#STR_DIS
	BLO	REV_ST		BLO	RT_TRN_ST
	JSR	STARFWD		JSR	STAROFF
	LDD	#0		LDD	#0
	STD	COUNT1		STD	COUNT2
REV_U_TRN	LDD	COUNT1	RT_TURN_LOOP	LDD	COUNT2
	CPD	#UTRN_DIS		CPD	#TRN_DIS
	BLO	REV_U_TRN		BLO	RT_TURN_LOOP
	JSR	INIT_FWD		JSR	INIT_FWD
	LDAA	RETURN		LDAA	RETURN
	BNE	BK_TRK_REV		BNE	BK_TRK_RT_TRN
	MOVE	#FWD,CRNT_STATE		MOVE	#FWD,CRNT_STATE
	BRA	REV_EXIT		BRA	RT_TRN_EXIT
BK_TRK_REV	JSR	INIT_FWD	BK_TRK_RT_TRN	MOVE	#BK_TRK,CRNT_STATE
	MOVE	#BK_TRK,CRNT_STATE			
REV_EXIT	RTS		RT_TRN_EXIT	RTS	

LT_TRN_ST	LDD	COUNT1
	CPD	#STR_DIS
	BLO	LT_TRN_ST
	JSR	PORTOFF
	LDD	#0
	STD	COUNT1
LT_TURN_LOOP	LDD	COUNT1
	CPD	#TRN_DIS
	BLO	LT_TURN_LOOP
	JSR	INIT_FWD
	LDAA	RETURN
	BNE	BK_TRK_LT_TRN
	MOVE	#FWD,CRNT_STATE
	BRA	LT_TRN_EXIT
BK_TRK_LT_TRN	MOVE	#BK_TRK,CRNT_STATE
LT_TRN_EXIT	RTS	

BK_TRK_ST	PULD	
	BRSET	PORTAD0,\$08,NO_BK_BUMP
	JSR	INIT_SBY
	MOVE	#SBY,CRNT_STATE
	JMP	BK_TRK_EXIT
NO_BK_BUMP	LDAA	NEXT_D
	BEQ	REG_PATHING
	BNE	IRREG_PATHING

REG_PATHING	LDAA	SENS_C
	BEQ	NO_RT_TRN
	PULA	
	PULA	
	STAA	NEXT_D
	JSR	INIT_RT_TRN
	MOVE	#RT_TRN,CRNT_STATE
	JMP	BK_TRK_EXIT
NO_RT_TRN	LDAA	SENS_B
	BEQ	RT_LINE_S
	LDAA	SENS_A
	BEQ	LEFT_TURN
	PULA	
	PULA	
	STAA	NEXT_D
	BRA	NO_LINE_S
LEFT_TURN	PULA	
	PULA	
	STAA	NEXT_D
	JSR	INIT_LT_TRN
	MOVE	#LT_TRN,CRNT_STATE
	JMP	BK_TRK_EXIT

IRREG_PATHING	LDAA	SENS_B
	BEQ	NO_LT_TRN
	PULA	
	STAA	NEXT_D
	JSR	INIT_LT_TRN
	MOVE	#LT_TRN,CRNT_STATE
	JMP	BK_TRK_EXIT
NO_LT_TRN	LDAA	SENS_C
	BEQ	RT_LINE_S
	LDAA	SENS_A
	BEQ	RIGHT_TURN
	PULA	
	PULA	
	STAA	NEXT_D
	BRA	NO_LINE_S
RIGHT_TURN	PULA	
	PULA	
	STAA	NEXT_D
	JSR	INIT_RT_TRN
	MOVE	#RT_TRN,CRNT_STATE
	JMP	BK_TRK_EXIT

Have a subroutine which follows the lines, which depends on the sensor. These functions simply cause the eebot to turn based on a set of conditions which is called in the main code. One of the subroutines checks the state of the led and updates the Dispatcher.

The rest code consists of utility subroutine functions which have been covered in the previous labs and in the Guider code.

Problems Encountered

Some of the main problems we encountered when working on the project is listed down below:

- Robot kept moving to the left
- One of the LED lights did not update/blink while the other ones did as expected
- Sometimes the LED stopped flashing after a while of running the bot

These are the main problems we encountered during our work on the project. We tried many different solutions to all these problems but none of them seemed to work. One of the possible scenarios was that we received a faulty bot where one of the sensors did not update or work as expected. Another problem could have been within the code portion of the project. Since the bot kept moving left and couldn't get a sense of following the black line, one possible explanation is that we coded the bot and it got stuck in a state of rotation. Evidently, with the bot not being able to switch states based off of the sensors, it stayed in rotation and couldn't respond to any stimulation.

Key Takeaways

Some key takeaways from this project we had is to have better time management of the software development and testing part of the project. Preparing different code blocks for our project ahead of time can allow us to test them altogether and figure out which ones are the most suitable for us. On top of that, we should have tested the code out more frequently instead of cramming it all in the last couple of days before our demo time. Another key takeaway is to be open to re-coding functions of previous labs. We had a mindset of keeping all the functions and code of the previous labs because they were running properly. As a result, we did not want to alter any lab 5 code until we kept troubleshooting the project and had no option but to change a few parts of that code. In hindsight, we should have been more open to looking at all possibilities of code alteration during the development and testing phase. This project was a great experience for the both of us, we learned a lot in terms of what it takes to properly manage and develop a software project. In the future, we would definitely start our work a few weeks prior to the due date and consistently test and debug our software while documenting both the bad and good parts of the code.